Learning Narrative Structure from Annotated Folktales

by

Mark Alan Finlayson

B.S.E., University of Michigan (1998)
S.M., Massachusetts Institute of Technology (2001)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2012

Author . . . .
Department of Electrical Engineering and Computer Science
December 20, 2011

Certified by . . . .
Patrick H. Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Chair of the Department Committee on Graduate Students

Learning Narrative Structure from Annotated Folktales
by
Mark Alan Finlayson

Submitted to the Department of Electrical Engineering and Computer Science
on December 20, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Narrative structure is an ubiquitous and intriguing phenomenon. By virtue of structure we recognize the presence of *Villainy* or *Revenge* in a story, even if that word is not actually present in the text. Narrative structure is an anvil for forging new artificial intelligence and machine learning techniques, and is a window into abstraction and conceptual learning as well as into culture and its influence on cognition. I advance our understanding of narrative structure by describing Analogical Story Merging (ASM), a new machine learning algorithm that can extract culturally-relevant plot patterns from sets of folktales. I demonstrate that ASM can learn a substantive portion of Vladimir Propp's influential theory of the structure of folktale plots.

The challenge was to take descriptions at one semantic level, namely, an event timeline as described in folktales, and abstract to the next higher level: structures such as *Villainy*, *Stuggle-Victory*, and *Reward*. ASM is based on Bayesian Model Merging, a technique for learning regular grammars. I demonstrate that, despite ASM's large search space, a carefully-tuned prior allows the algorithm to converge, and furthermore it reproduces Propp's categories with a chance-adjusted Rand index of 0.511 to 0.714. Three important categories are identified with F-measures above 0.8.

The data are 15 Russian folktales, comprising 18,862 words, a subset of Propp's original tales. This subset was annotated for 18 aspects of meaning by 12 annotators using the Story Workbench, a general text-annotation tool I developed for this work. Each aspect was doubly-annotated and adjudicated at inter-annotator F-measures that cluster around 0.7 to 0.8. It is the largest, most deeply-annotated narrative corpus assembled to date.

The work has significance far beyond folktales. First, it points the way toward important applications in many domains, including information retrieval, persuasion and negotiation, natural language understanding and generation, and computational creativity. Second, abstraction from natural language semantics is a skill that underlies many cognitive tasks, and so this work provides insight into those processes. Finally, the work opens the door to a computational understanding of cultural influences on cognition and understanding cultural differences as captured in stories.

Dissertation Supervisor:

Patrick H. Winston
Professor, Electrical Engineering and Computer Science

Dissertation Committee:

Whitman A. Richards
Professor, Brain & Cognitive Sciences

Peter Szolovits
Professor, Electrical Engineering and Computer Science &
Harvard-MIT Division of Health Sciences and Technology

Joshua B. Tenenbaum
Professor, Brain & Cognitive Sciences

to LDF

Acknowledgments

Without generous funding by the government this work would not have been possible. My initial explorations of intermediate-feature-based analogical retrieval were supported by National Science Foundation grant IIS-0218861, “Reinventing Artificial Intelligence.” It was then that I began to consider the challenge of breaking stories into their constituents. Development of the Story Workbench was begun under National Science Foundation grant IIS-0413206, “Representationally Complete Analogical Reasoning Systems and Steps Toward Their Application in Political Science.” I co-wrote that proposal with Patrick Winston, my advisor. It was the first of the many deeply educational experiences he provided me over my graduate career. He was an armamentarium of sage advice, and he was surely a *sine qua non* of this work and my doctorate.

The Air Force Office of Scientific Research, under contract A9550-05-1-032, “Computational Models for Belief Revision, Group Decisions, and Cultural Shifts,” supported further development of the Story Workbench as well as the early implementations of Analogical Story Merging. The AFOSR funding was particularly influential on my work, primarily via Whitman Richards, who early on pointed out to me the importance of stories in culture, and guided my work in that direction; he later joined my committee and, as attested by the excellent advice and support he provided, became my *de facto* co-advisor.

The Defense Advanced Research Projects Agency, under contract FA8750-10-1-0076, “Defining and Demonstrating Capabilities for Experience-Based Narrative Memory,” provided the support for annotation of the Russian folktales that are the data underlying this work. The remainder of my graduate career, including the final experiments and writing of the dissertation itself, was supported by the Office of the Secretary of Defense, under Minerva contract N00014-09-1-0597, “Explorations in Cyber-International Relations.” Without all of these various funds this work would have been impossible to complete, even within the generous timeline it has so far consumed.

Brett van Zuiden, my especially bright and motivated undergraduate research assistant during the Summer and Fall of 2010, was responsible for a significant portion of the implementation described in Section 3.4.3.

Finally, my annotation team was critical to the success of this work. They were Julia Arnous, Saam Batmanghelidj, Paul Harold, Beryl Lipton, Matt Lord, Sharon Mozgai, Patrick Rich, Jared Sprague, Courtney Thomas, Geneva Trotter, and Ashley Turner.

Contents

List of Figures

List of Tables

Chapter 1

Motivation

If you have never read the folktales of another culture, you should: it is an educational exercise. It doesn't take many tales — ten, perhaps twenty, an hour or two of reading at most — before you will begin to sense the repetition, see a formula. Perhaps the first parallel to jump out are those characters that make multiple appearances: *Another tale about the adventures of this hero?*; *This is the same villain as before!* An idiosyncratic phrase will be repeated: *Why do they say it that way, how funny!* The plots of the tales will begin to fall into groups: *I see, another dragon-kidnaps-princess tale.* You will notice similarities to the tales you heard as a child: *This is just like Cinderella.* Other things will happen time and again, but just not make sense: *Why in the world did he just do that?!*

Such formulaic repetition, or *narrative structure*, has been the subject of much speculation. Folktales drawn from a single culture, or group of related cultures, seem to be built upon a small set of patterns, using only a small set of constituents. Such seemingly arbitrary restriction in form and content is at the very least intriguing; it is is especially interesting in light of mankind's ability to tell almost any kind of story with infinite variation of setting, character, plot, and phrasing. Why are folktales so restricted, and what does it mean? Perhaps the most interesting and compelling hypothesis is that these restrictions reflect the culture of the tales from which they are drawn. This hypothesis holds that stories passed down through oral traditions — folktales — are subject to a kind of natural selection, where folktale constituents that are congruent with the culture are retained and amplified, and constituents that are at odds with the culture are discarded, mutated, or forgotten. Therefore, according to this hypothesis, folktales encode cultural information that includes, for example, the basic types of people (heros, villains, princesses), the good and bad things that happen, or what constitutes heroic behavior and its rewards.

The first step in investigating such an intriguing hypothesis is delineating and describing the narrative structure found in a particular set of tales. In 1928 Vladimir Propp, a Russian folklorist from St. Petersburg, made a giant leap toward this goal when he published a monograph titled *Morfólogija skázki*, or, as it is commonly known in English, *The Morphology of the Folktale* (Propp 1968). Within it he described a laborious analysis of one hundred Russian fairy tales that culminated in 31 plot pieces, or *functions*, out of which Propp asserted you could build any one of the one hundred tales. He identified seven character types, *dramatis personae*, that populated the tales. And he specified what is, in essence, a grammar of piece combination, describing how pieces occur in particular orders, groups, and combinations. Together, these functions, *dramatis personae*, and rules of combination formed what he called a *morphology*. It was a seminal work that transformed our view of folktales, culture, and cultural cognition.

The primary, concrete contribution of this work is the demonstration of Analogical Story Merging (ASM), a procedure that can learn a substantive portion of Propp's analysis from actual tales. The data on which the algorithm was run comprised 15 of Propp's simplest fairy tales translated into English, totalling 18,862 words. I supplied to the algorithm much of the same information that a person would consider "obvious" when reading a folktale. I call this information the *surface semantics* of the tale, and it includes, for example, the timeline of the events of the story, who the characters are, what actions they participate in, and the rough semantics of words in the text. To capture this information I created the *Story Workbench*, an annotation tool that allowed a team of 12 annotators to mark up, semi-automatically, 18 aspects of the surface semantics of the tales. This tool allowed me to simultaneously take advantage of state-of-the-art automatic techniques for natural language processing, while avoiding those techniques' limitations and shortcomings.

Why is this work an important advance? It is important for at least three reasons. First, it is a novel application of machine learning, artificial intelligence, and computational linguistics methods that points the way forward to many potential new technologies. It suggests, for example, how we might finally achieve the long-held goal of information retrieval by concept, rather than by keyword. It illustrates how we might one day endow computers with the ability to discover complex concepts, a necessary skill if computers are to one day think creatively. It demonstrates a foundation on which we might build tools for understanding and engineering culture: knowing which parts of stories will resonate with a culture will help us predict when ideas will catch fire in a population, giving rise to "revolution detectors." Understanding the structure of culture as captured in stories could allow us to diagnose the culture of an individual or a group, leading to more persuasive and effective negotiations, arguments, or marketing.

Second, this work advances our understanding of cognition, and in particular, the human ability to discover abstractions from complex natural language semantics. A person need not intend to see the structure in a set of folktales. After enough exposure, it simply appears. With additional effort and attention, a folklorist can extract even more insightful and meaningful patterns. This ability to discover implicit patterns in natural language is an important human ability, and this work provides one model of our ability to do this on both an conscious and unconscious level.

Third, this work advances our understanding of culture. Culture is a major source of knowledge, belief, and motivation, and so deeply influences human action and thought. This work provides a scientific, reproducible foundation for folktale morphologies, as well as a scientific framework for understanding the structure of cultural knowledge. Understanding culture is important not only for cognitive science and psychology, but is the ultimate aim of much of social and humanistic scholarship. It is not an exaggeration to say that we will never truly understand human action and belief — indeed, what it means to be human — if we do not understand what culture is, what it comprises, and how it impacts how we think. This work is a step in that direction.

1.1 Outline

The plan of attack is as follows. I devote the remainder of this chapter to expanding the above motivations. In Chapter 2 I detail Propp's morphology and its features relevant to the current study. In Chapter 3 I describe the basic idea of the Analogical Story Merging (ASM) algorithm as well as three proof-of-concept experiments that demonstrated the algorithm's feasibility. In Chapter 4 I explain the data I collected, the surface semantics of the 15 folktales, by describing the 18 different aspects of meaning that were annotated on the texts, along with the Story Workbench annotation tool and the annotation procedures. Chapter 5 is the centerpiece of the work, where I show how I converted the annotations into grist for the ASM algorithm, and how I transformed Propp's theory into a gold standard against which the algorithm's output could be compared. Importantly I characterize the result of running ASM over the data both quantitatively and qualitatively. Chapter 6 contains a review of related work during which I outline alternative approaches. I close in Chapter 7 with a succinct list of the four main contributions of the work.

1.2 Artificial Intelligence & Machine Learning

There are at least three reasons why this work is important. First, it is important is because it points the way forward to a host of important applications in a number of domains related to artificial intelligence and machine learning.

Information Retrieval This work may lead to search engines that index by concept instead of keyword. A filmgoer could ask for "a movie as dramatic" as the one you saw last week, an intelligence analyst could query for "battles like the Tet offensive" from a military database, or a doctor could find "similar case histories" in the hospital's records. In other words, the computer would be able to retrieve like an expert, rather than a novice (Finlayson and Winston 2006).

Concept Learning In a related vein, this work may one day endow computers with the ability to discover complex concepts on their own, a necessary skill for creative thinking. Imagine a computer extracting from reports of criminal activity a set of likely *modi operandi*, or from a set of seemingly contradictory appeals court cases a set of legal principles that explain the overarching framework. In other words, the computer would able to find, as a human expert would, the key features, groupings, and explanations in a set of data. Such an ability has yet to be reproduced in a computer, but it is clearly necessary if we are ever to achieve true artificial intelligence.

Persuasion & Negotiation A precise understanding of cultural information encoded in stories could allow the diagnosis of culture in an individual or group. Imagine that a person tells the computer a few stories, and receives in return an analysis of their cultural make-up, say, one-quarter this culture, three-quarters that culture. This would be useful for adjusting the form and content of messages directed to that person. It could be used to improve the outcome of negotiations by indicating which arguments would be most effective. It might be applied to increase sales or brand appeal by guiding advertising or marketing strategies. It could increase our ability to defend against political propaganda. It may allow us to know which ideas and stories will "catch fire" in a particular population, leading to an ability to detect revolutions before they begin.

Computational Humanities This work may give us the foundation for what would be a whole new field: *Cultural Genomics*. If we can reliably extract the "DNA" of a set of stories, in the form of its folktale morphology, we could precisely arrange cultures into relationships that, until now, have only been intuitive. In such a field, *Comparative Cultural Genomics*, the familial relationships between cultures could be laid bare in the same scientific way that the relationships between species and languages have been. Historical data opens the way, similarly, to *Cultural Phylogenomics*, in which we would be able to trace the lineage of the world's cultures.

1.3 Cognition & Abstraction

Second, this work is important because it advances our understanding of the human ability to discover abstractions, at both the conscious and unconscious level, in complex natural language stimuli.

At the conscious level, we have endeavors such as Propp himself undertook when devising his morphology. This type of effort is deliberative, where the stimuli are gathered for comparing and contrasting, aimed at identify their commonalities, differences, and patterns. The process is time consuming and labor intensive, and requires cognitive aids in the form of written notes, tables of figures, graphs of relationships, and so forth. Propp himself describes his process in the foreword of his monograph:

> The present study was the result of much painstaking labor. Such comparisons demand a certain amount of patience on the part of the investigator. ... At first it was a broad investigation, with a large number of tables, charts, and analyses. It proved impossible to publish such a work, if for no other reason than its great bulk. ... An attempt at abbreviation was undertaken ... Large comparative charts were excluded, of which only headings remain in the appendix. (Propp 1968, p.xxv)

On the other hand, at an unconscious level we often see abstract similarities with little effort. This has been shown in the experimental psychology literature; for example, Seifert *et al.* (1986) showed that thematically similar episodes are often linked in memory. Thematically similar stories share higher-level plan-goal structure, but do not share objects, characters, or events. For example, two thematically-similar stories might be:

> Dr. Popoff knew that his graduate student Mike was unhappy with the research facilities available in his department. Mike had requested new equipment on several occasions,

> but Dr. Popoff always denied Mike's requests. One day, Dr. Popoff found out that Mike had been accepted to study at a rival university. Not wanting to lose a good student, Dr. Popoff hurriedly offered Mike lots of new research equipment. But by then, Mike had already decided to transfer. (Seifert et al. 1986, p.221)

> Phil was in love with his secretary and was well aware that she wanted to marry him. However, Phil was afraid of responsibility, so he kept dating others and made up excuses to postpone the wedding. Finally, his secretary got fed up, began dating, and fell in love with an accountant. When Phil found out, he went to her and proposed marriage, showing her the ring he had bought. But by that time, his secretary was already planning her honeymoon with the accountant. (Seifert et al. 1986, p.221)

These two stories can be said to each illustrate the theme of "Closing the barn door after the horse is gone." When experimental subjects were known to have compared the story similarities in their reading strategies, thematically-similar stories generated priming effects without inducing actual recall of a similar story. A similar effect was shown for analogically-related problems from different domains (Schunn and Dunbar 1996). These results strongly suggest that thematic representations can be unconsciously extracted and stored in memory.

Finding abstractions is a key cognitive skill, at both a conscious and unconscious level, one found across many human endeavors. Therefore this work is a step toward modeling this process, giving insight into the underlying processes of human cognition.

1.4 Culture & Folktales

Finally, this work is important because it sheds light on culture, and helps us understand how culture impacts, and is impacted by, cognition.

Culture is an indispensable part of the human experience: its artifacts fill our lives, its morals color our reactions, and its beliefs shape our actions. For the purposes of this work, culture is best considered to be a set of knowledge structures held in common by a particular group or category of people. Helen Spencer-Oatey puts it nicely:

> Culture is a fuzzy set of attitudes, beliefs, behavioural norms, and basic assumptions and values that are shared by a group of people, and that influence each member's behaviour and his/her interpretations of the 'meaning' of other people's behaviour. (Spencer-Oatey 2000, p.4)

Another definition, friendly to a computer science and cognitive science perspective, is that of Geert Hofstede:

> Culture is the collective programming of the mind which distinguishes the members of one category of people from another. (Hofstede 1980, p.21)

So culture is knowledge, and the folktale contributes to culture as an artifact that reflects and contains cultural knowledge.

A folktale is any traditional, dramatic, and originally oral narrative. Folktales include religious myths, tales for entertainment, purportedly factual accounts of historical events, moralistic fables, and legends. They can be contrasted with other sorts of *expressive culture* such as ritual, music, graphic and plastic art, or dance, or other aspects of so-called *practical culture* such as technology or economy (Fischer 1963). Because of the popularity of the hypothesis that they are a window onto cultural knowledge, folktales have been of central importance to those studying culture. Indeed, every major division of social and cultural anthropology has found the folktale useful for their purposes. Tylor, Frazer and Lévy-Bruhl (the early evolutionists), for example, were interested in folktales as examples of "primitive" reasoning processes. Malinowski (functionalism) emphasized myths, and said they provided a "charter" for society, justifying institutionalized behavior. Leach says that the cognitive function of myth is to present a model of the social structure. Bascom believed folktales figured importantly in inspiring and controlling members for the good of the society. Among social scientists, a popular view has been that the cognitive function of myth and folktales is to enculturate individuals to the social structure of the society (Fischer 1963).

Under the hypothesis that folktales are a windows into culture, culture gets “baked into” the tales through repeated retellings. Barlett (1920, 1932), in his seminal study of the transmission of oral tales, proposed six principles of transmission. The first three were principles of omission, in particular omission of the *irrelevant*, the *unfamiliar*, and the *unpleasant*. Via these, narrators omit parts of the story (a) that don’t make sense or don’t fit into the perceived causal structure (irrelevant), (b) with which they have little contact in their daily lives (unfamiliar), or (c) that cause distress or embarrassment (unpleasant). The second three mechanisms were principles of transformation, specifically *familiarization*, *rationalization*, and *dominance*. With these stories are changed by (a) replacing the unfamiliar with the familiar (familiarization), (b) inserting reasons and causes where none were before or changing consequences to something that makes sense (rationalization), and (c) emphasizing some part that seems especially salient to the teller, while downplaying other parts (dominance). Fischer emphasizes this view with a nicely technological metaphor:

> The process of development and maintenance of a folktale involve a repeated cycle of transmission and reception similar to that involved in a radio message. The static in a radio message may be compared to the distortion and selective elimination and emphasis to which the ‘original stimuli’ of a folktale are subject in human brains. ... We should recall, moreover, that folktales ... are social products and that in the long run only what is significant to a sizable segment of the society will be retained in a tale. (Fischer 1963, p.248)

The hypothesis, therefore, is that folktales strongly reflect cultural knowledge because culture changes them during transmission, and that is why they display such a uniformity of form and content. If this were so, then it would be of wide-ranging interest if the narrative structure of a culture’s folktales could reliably be exposed to scientific investigation. If these narrative structures have psychological reality, they potentially would be a powerful window into cultural thought. Propp’s theory of the morphology is one of most precise formulations of the narrative structure of folktales to date, and thus presents the most compelling target of study. Until now the extraction of morphologies has remained a manual task, the purview of anthropological virtuosos (see Section 6.1.5). Constructing a morphology for a particular culture takes many years of reading and analysis. Once complete, it is unclear how much the morphology owes to the folklorist’s personal biases or familiarity with other extant morphologies, rather than truly reflecting the character of the tales under investigation. Furthermore, reproduction or validation of a morphological analysis is a time-consuming, prohibitively difficult endeavor. Thus this work is a vital step toward bringing morphologies, and therefore culture, into the realm of repeatable, scientific inquiry.

Chapter 2

Propp's Morphology

The primary contribution of this work is a procedure that can learn a substantive portion of Propp's morphology from actual Russian folktales. This chapter details the relevant features of Propp's morphology and outlines my strategy for learning the morphology from data. I first describe the structure of Propp's morphology, which can be broken into three levels, each with its own grammatical complexity. Given this analysis, I outline a learning strategy informed by the state of the art in grammar inference. Finally, I review the folktales from which Propp derived his morphology and discuss how I chose the specific folktales that I used as data.

Propp's morphology captures regularities in the folktales he examined, but what sort of regularities? Folktales host a range of types of repetition, including repetition of words, phrases, classes of person or object, plot sequences, and themes. Although Propp acknowledged many types of repetition, he focused on primarily repeated plot elements, which he called *functions*, and their associated character roles, or *dramatis personae*.

Propp offered the following definition for functions: "Function is understood as an act of a character, defined from the point of view of its significance for the course of the action." (Propp 1968, p.21) Propp identified 31 major types of functions; examples include *Villainy*, *Struggle*, *Victory*, and *Reward*. Importantly, he held that functions defined "what" was happening, but did not necessarily specify "how" it happened — that is, functions could be instantiated in many different ways. Each function involved a set of characters who filled certain roles, which Propp called the *dramatis personae* of the morphology. He identified seven *dramatis personae* classes: *Hero*, *Villain*, *Princess*, *Dispatcher*, *Donor*, *Helper*, and *False Hero*.

Because of the primacy of functions in Propp's theory, I focus on learning the functions categories and their arrangement, and leave the discovery of other types of repetition as future work.

2.1 Three Levels

Propp's morphology comprises three levels: gross structure (moves), intermediate structure (functions), and fine structure (subtypes). A tale is made up of moves. A **move** is a "rudimentary tale" made up of functions. A **function** is a plot element that has a major type that determines its position, purpose, and *dramatis personae*. Each function has a minor type, or **subtype**, the choice of which may impose constraints on subtypes of other functions in the tale.

2.1.1 Gross Tale Structure: Moves

Propp defined the top-level structure of a tale to be an optional preparatory sequence, followed by some number of moves, possibly intermingled. The grammatical complexity of this level is at least context-free. Propp identified a *move* as a plot development that proceeds from a motivating complication function — in Propp's case, a villainy or a lack — through other intermediate functions to an eventual liquidation of that conflict. A move may conclude with a set of dénoument functions.

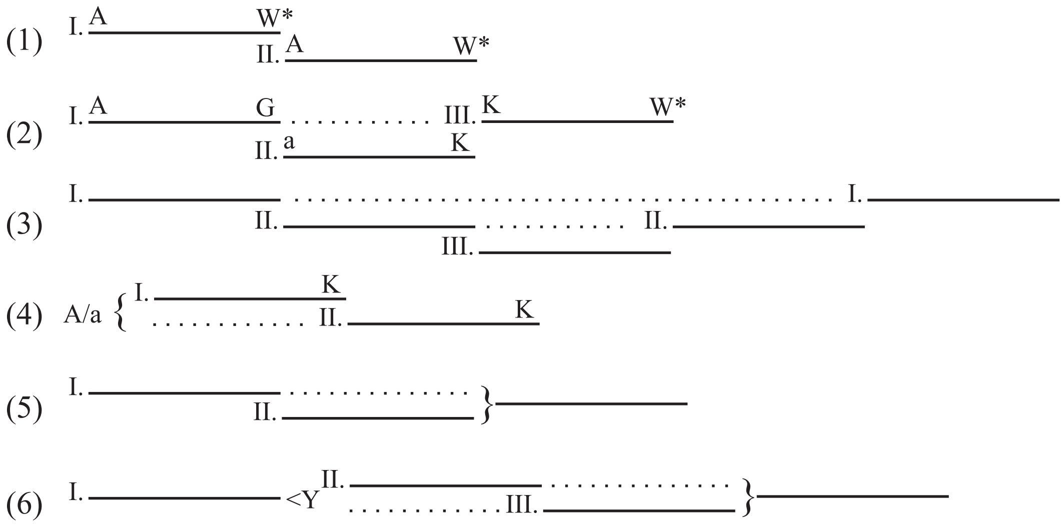


Figure 2-1: Six combinations of moves identified by Propp. Moves are numbered with roman numerals, and letters indicate starting and ending functions. A dashed line means an interrupted move. (1) Concatenation. (2) Interruption. (3) Nested interruption. (4) Two simultaneous conflicts liquidated in separate, concatenated moves. (5) Two moves with a common ending. (6) Combination of the first and fifth, incorporating leave-taking at a road marker.

Non-Terminal	Description	Terminal	Description
T	full tale	π	preparatory sequence
U	tale *sans* preparation	κ	complication
M_n	move number n	μ	first part of a move
		ν	second part of a move
		λ	liquidation

Table 2.1: Inferred non-terminals (left) and terminals (left) of Propp's move level.

The simplest tales are those containing only a single move. Moves in a multi-move tales, on the other hand, may intermingle in complex ways. Propp identified six ways moves could combine to make a tale; these are illustrated in Figure 2-1. These combinations are straightforward. First is *concatenation*, in which whole moves are simply appended to the end of the last move. Second is *interruption*, in which a move is interrupted by a complete other move. Third is a nested interruption, which indicates episodes themselves may be interrupted[1]. Fourth is a tale that begins with two conflicts, the first of which is liquidated completely by the first move before the second conflict is liquidated by the second move. Fifth, two moves may have a common ending. Finally, sixth, the first and fifth structures may combine.

Propp did not propose production rules to describe these combinations; indeed, such a formalism did not yet exist. Nevertheless, we can infer a rough set of rules, which are laid out in Tables 2.1 and 2.2. Clearly these rules define at least a context-free grammar, an observation consistent with Lakoff's (1972) analysis.

2.1.2 Intermediate Tale Structure: Functions

The intermediate level was Propp's most important, and, critically, it is a regular grammar. It outlined the identity of the functions, their *dramatis personae*, and their sequence. He identified 31

[1] There seems to be a typographical error in Propp's monograph (Propp 1968, p.93), in which in the third structure, the first move is erroneously continued in the third position.

Rule	Description
$T \rightarrow \pi U$	tales may have a preparatory sequence
$T \rightarrow U$	tales may lack a preparatory sequence
$Y \rightarrow MU$	moves may be directly concatenated (1)
$M \rightarrow \kappa\mu\nu\lambda$	moves are begun by a complication and ended by a liquidation
$M_n \rightarrow \kappa_n\mu_n M_m \nu_n$	moves may be interrupted by another move (2 & 3)
$U \rightarrow \kappa_n\kappa_m\mu_n\nu_n\lambda_n\mu_m\nu_m\lambda_m$	tales may have two complications that are sequentially resolved (4)
$U \rightarrow \kappa_n\mu_n\nu_n\kappa_m\lambda_m\nu_{n,m}$	two moves may have a common ending (5)

Table 2.2: Inferred rules of Propp's move level. The numbers in parentheses indicate move combinations explicitly identified by Propp, as illustrated in Figure 2-1.

major types of function, listed in Table 2.3.

Roughly speaking, when a function appears, it appears in alphabetical order as named in the table. This can be seen by examining Propp's Appendix III, wherein he identifies the presence and order of the functions in 45 of his 100 tales (Propp 1968, p.133).

There are multiple pairs of functions that are co-dependent. Propp singled out the pairs $H - I$ (fight and victory) and M-N (difficult task and solution) as defining of four broad classes of tales: H-I present only, M-N present only, both pairs present, or neither present. In Propp's observation the first two classes predominant. Other functions that nearly always occur together are γ and δ (interdiction and violation), ϵ and ζ (reconnaissance and information receipt), and η and θ (trick and trick success), and Pr and Rs (pursuit and rescue).

There are several exceptions to the alphabetical-order rule. First, encounters with the donor, instantiated in functions D, E, and F, may occur before the complication. This is reflected explicitly in Propp's Appendix III, where he provides a space for D, E, and F to be marked before the complication.

Second, certain pairs may also participate, rarely, in order inversions, for example, when a Pr-Rs pair comes before an H-I pair. Propp merely noted these inversions, he did not explain them. This is one of the most dissatisfying aspects of Propp's theory. However, because no inversions of this type occur in my data, I will ignore this problem for the remainder of this work.

Finally, Propp notes a phenomenon he calls *trebling*. In it, a sequence of functions occurs three times rather than just once. One classic example is the tale *The Swan Geese*. In it, a little girl is tasked by her parents to watch over her younger brother. The girl forgets and goes out to play, and the Swan Geese kidnap the unwatched brother. In the ensuing chase, the daughter encounters three donors (a stove, an apple tree, and a river) who respond negatively to her insolent query (a trebled negation of DEF). On the return with the rescued brother, she encounters each again and is successful in securing their help (a second trebling). Such a phenomenon can be easily incorporated into a regular grammar by replacing each `?` with `[0-3]`.

The grammar inferred from these observations is laid out in Tables 2.4 and 2.5.

2.1.3 Fine Tale Structure: Subtypes

The most detailed level of Propp's morphology is what I refer to as the *subtype* level. For each function Propp identified, he also identified specific ways, in actual tales, that the function was instantiated. The first function, β, illustrates the idea. This function is referred to as *Absentation*, and was defined by Propp as "One member of a family absents himself from home." Propp proposed three subtypes of β: (1) The person absenting himself can be a member of the older generation, (2) an intensified form of absentation is represented by the death of parents, and (3) members of the younger generation absent themselves. Functions range from having no subtypes, in the case of simple atomic actions such as C, "decision to counteract", to nearly twenty in case of A, "villainy". The subtype level adds additional complexity to the function level, but can be incorporated into the function regular grammar or move context-free grammar in the form of a feature grammar or generalized phrase structure grammar (GPSG).

Symbol	Name	Key *dramatis personae*
β	Absentation	Hero
γ	Interdiction	Hero
δ	Violation	Hero
ϵ	Reconnaissance	Villain
ζ	Delivery	Hero
η	Trickery	Villain
θ	Complicity	Hero
A/a	Villainy/Lack	Villain, Dispatcher, Princess
B	Mediation	Dispatcher, Hero, Helper
C	Beginning Counteraction	Hero, Helper
$\uparrow$	Departure	Hero, Helper
D	Donor Encounter	Hero, Helper, Donor
E	Hero's Reaction	Hero, Donor
F	Receipt of Magical Agent	Hero, Donor
G	Transference	Hero, Helper
H	Struggle	Hero, Helper, Villain
I	Victory	Hero, Helper, Villain
J	Branding	Hero, Princess
K	Tension Liquidated	Hero, Helper, Princess
$\downarrow$	Return	Hero, Helper, Princess
Pr	Pursuit	Hero, Helper, Princess, Villain
Rs	Rescue	Hero, Helper, Princess, Villain
o	Unrecognized Arrival	False Hero, Dispatcher, Princess
L	Unfounded Claims	False Hero, Hero, Dispatcher, Princess
M	Difficult Task	Hero, Helper
N	Solution	Hero, Helper
Q	Recognition	Hero, Princess
Ex	Exposure	False Hero
T	Transfiguration	Hero
U	Punishment	False Hero
W	Reward	Hero, Helper, Dispatcher

Table 2.3: Propp's 31 functions, the terminals of the grammar describing Propp's function level.

Non-Terminal	Description
Π	preparatory sequence (see Table 2.1)
M_n	move number n (see Table 2.1)
Δ	encounter with a donor
Φ	fight and victory
Σ	difficult task and solution

Table 2.4: Inferred non-terminals of Propp's function level.

Rule	Description
$\Pi \rightarrow \alpha?\beta(\gamma\delta)?(\epsilon\zeta)?(\eta\theta)?$	pairs of preparation functions occur together
$M_n \rightarrow \Delta?AB?C?\Delta?G?...$	functions usually occur in alphabetical order
$\Delta \rightarrow (DE?F?)\|(D?EF?)\|(D?E?F)$	donor encounter
$\Sigma \rightarrow MN$	difficult task/solution pair
$\Phi \rightarrow (HI?)\|(H?I)$	struggle/victory pair

Table 2.5: Inferred rules of Propp's function level.

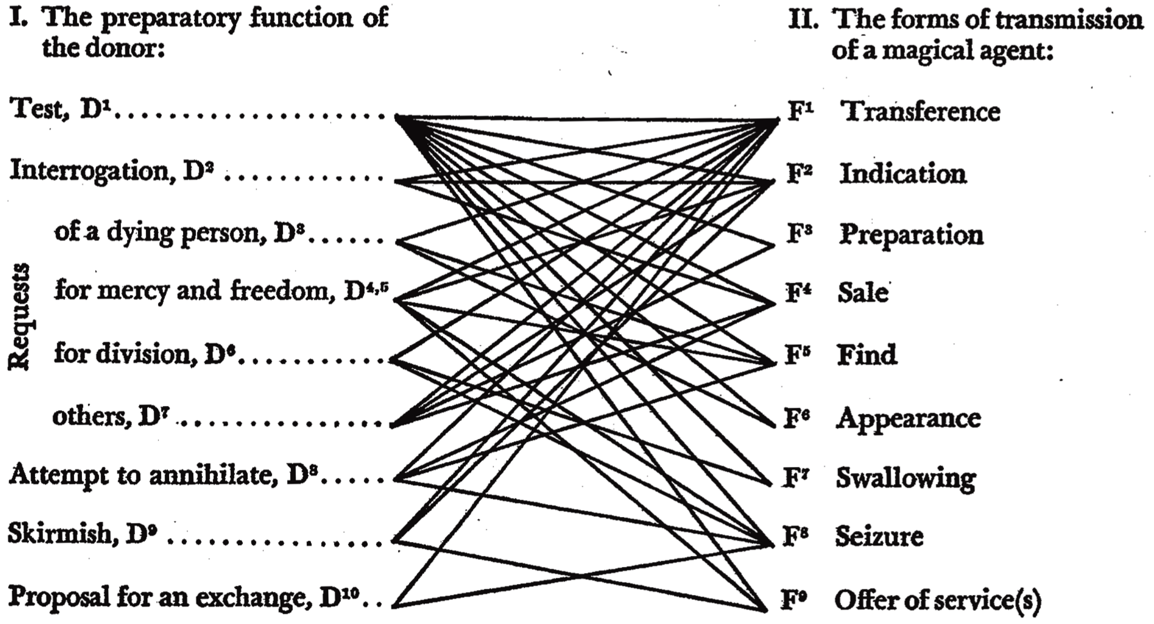

I. The preparatory function of the donor:
II. The forms of transmission of a magical agent:
Test, D^{1}
Interrogation, D^{2}
of a dying person, D^{3}
for mercy and freedom, $D^{4,5}$
for division, D^{6}
others, D^{7}
Requests
Attempt to annihilate, D^{8}
Skirmish, D^{9}
Proposal for an exchange, D^{10}
F^{1} Transference
F^{2} Indication
F^{3} Preparation
F^{4} Sale
F^{5} Find
F^{6} Appearance
F^{7} Swallowing
F^{8} Seizure
F^{9} Offer of service(s)

Figure 2-2: Propp's figure (Propp 1968, Figure 1) indicating how subtypes of F depend on subtypes on D.

By Propp's own admission, these subtype groups were somewhat *ad hoc*. They are interesting for two reasons. First, they address the specifics of how a function is instantiated in a tale, beyond whether or not it is present. Because the number of subtypes for most of the functions is rather small, the semantic character of events should be an important feature when classifying events into functions.

Second, subtypes are interesting because of the long-range dependencies in which they participate. If a particular subtype is chosen early in the tale, this can require one to choose a specific subtype of a much later function. Propp's primary example of this is how the forms of transmission of the magical item (F) depend on the initial encounter with the donor (D), as illustrated in Figure 2-2. To a certain extent these constraints reflect commonsense restrictions. An illustrative example of this is the M-N pair, difficult task paired with a solution. If the difficult task is to retrieve a certain magical item (and not, for example, to kill a certain monster), the task is naturally solved by finding the item and not by killing a monster.

2.2 Learning Strategy

Having determined the grammatical complexity of the different levels of Propp's morphology, I formulate a learning strategy for the work.

2.2.1 Grammatical Inference

The learning strategy must be informed by the state of the art in grammatical inference. Results from that area give a number of bounds and limits on what can be achieved. This effort falls into the category of "learning a grammar from text," with the additional complication that the technique must learn not just the grammatical rules, but the alphabet of the grammar as well. *Learning from text* means that the learning technique is limited to observations of positive examples. The data provide no negative evidence, no counter-examples, and no additional information from Oracles or other sources commonly leveraged to achieve grammar learning (Higuera 2010).

The first difficulty to keep in mind is that the more powerful the grammar, the more difficult it is to learn. Furthermore, the learning techniques used for one type of grammar will not be able, merely representationally speaking, to learn a more powerful grammar. Therefore, one must carefully select the grammar learning technique so as not to preclude learning a grammar that is at least as powerful as the target.

However, the primary result that restricts our potential is Gold's original result showing that any class of grammars that includes all finite languages, and at least one infinite language, is not *identifiable in the limit* (Gold 1967). This means that all the classes of grammars under consideration here, including regular, context-free, and more powerful grammars, are not identifiable in the limit. The intuition behind this result is that, while trying to learn a grammar solely from positive examples, if one forms a hypothesis as to a specific form of the grammar consistent with the examples seen so far and that hypothesis erroneously covers a negative example, no number of positive examples will ever stimulate one to remove that negative example from the hypothesis.

There are several ways around this problem. First is to restrict the class of the language being learned to something that does not include all finite languages. These include classes such as k-testable languages, look-ahead languages, pattern languages, and planar languages (Higuera 2010). Propp's morphology, considered all three levels together, clearly does not fall into one of these classes, as they are all less expressive than a context-sensitive language. The extent to which Propp's intermediate level (the level in which we are particularly interested) falls into one of these classes is not at all clear. In any case no formal result treats the case where a language is in one of these classes *and* one must also simultaneously learn the alphabet.

A second way around Gold's result is to admit additional information, such as providing negative examples, querying an Oracle for additional information about the grammar or language, or imposing some pre-determined bias into the learning algorithm. While negative examples or an Oracle are not consistent with the spirit of this work, the idea of introducing a bias is perfectly compatible. It is highly likely that Propp himself had cognitive biases, unconscious or not, for finding certain sorts of patterns and similarity.

A third option around Gold's result is to change the requirement of identification in the limit. One such approach is called *probably-approximately correct* (PAC) learning, where the learned grammar is expected to approach closely, but perhaps not identify exactly, the target grammar (Valiant 1984). My strategy combined the second and third options: I defined a learning bias in the algorithm and did not strive for exact identification to measure success.

2.2.2 Learning Strategy for Propp's Data

The grammatical complexity of all three levels of Propp's morphology taken together is at least that of a feature grammar or generalized phrase structure grammar. Because my goal was to learn both the grammar rules and the identities of the grammar symbols themselves, learning the whole of Propp's morphology becomes a complex hierarchical learning problem involving grammatical rules, non-terminals, and terminals at each of the three levels. This means that to tackle Propp's full grammar immediately, I would have had to engage the most powerful grammar learning techniques available, and supplement them with additional potent techniques. This seemed infeasible, and so, for this first attempt at learning Propp's morphology from the stories themselves, I limited my goals to make the problem tractable and deferred several difficult parts of the problem to future work.

The key observation was that the most important part of Propp's morphology is the functions. Moves are defined by virtue of their constituent functions, and subtypes are a modulation on functions. It is in the identities of the functions, and their arrangement, that Propp has had a major impact. Indeed, functions lies at the intersection of cultural specificity and cultural uniformity, and the majority of work that has built upon Propp has focused on this intermediate level (Díaz-Agudo, Gervás, and Peinado 2004; Halpin, Moore, and Robertson 2004).

Therefore, I simplified the problem by focusing on learning the regular grammar and categories of the function level, and I deferred the learning of both the move and subtype levels. As I show in Chapter 5, I also deferred learning the *dramatis personae* classes.

Deferring learning the move level was relatively straightforward. Propp manually identified the number of moves in each of the tales he analyzed. Using this analysis as ground truth, I excluded all multi-move tales from the data, thus eliminating the move level as a variable.

To defer the subtype level I made use of the fact that while there are a wide range of predicates that may instantiate a particular function, they are often semantically quite closely related even across function subtypes. Therefore, it was plausible to merge together these variations and not attempt to learn subtype categories or dependencies.

Deferring the *dramatis personae* classes was also straightforward. I defined an annotation scheme, described in Chapter 4, which allowed my annotators to mark characters that filled the role of a particular *dramatis personae*.

Therefore, the learning strategy was first to filter out variations at the move level by restricting the data to tales Propp identified as single move, second, to ignore variations at the fine level by using sufficiently coarse semantic similarity measures, and, third, to explicitly mark *dramatis personae* classes. Remaining were the function categories and the regular grammar of the intermediate level. Learning regular grammars (with known symbols) is a problem on which there has been some traction, and I leveraged that work to construct a new algorithm for learning the symbols simultaneously with the grammar. This is described in outline in the next chapter, and in detail in Chapter 5.

2.3 Texts and Translations

With a learning strategy in hand, it remains to consider the data used to implement that strategy.

Collecting, selecting, and preparing folktales for analysis is a task fraught with complexity (Fischer 1963). The collector's rapport with the narrator affects the types of tales told: when the relationship is new, tales are biased toward lighter, entertaining fare, whereas a more established, trusting relationship encourages the transmission of more serious and important stories. The season of collection can be important, as different stories are activated at different times of the year. The duration of collection affects the sample size and variability. A few days of collecting may be easy, but the sample will be impoverished relative to a collection assembled over months or years. Indeed, even the personality of the collector is important, as skilled narrators tailor their stories to the audience and their reactions. Once a tale has been recorded, care must be taken to prevent its corruption in written form. As soon as a tale is written down, it is possible to revise and to put quite a bit of thought and time into the text, unlike during an oral recitation. Once a collection has been assembled, how should one select the tales for analysis? There may be hundreds or thousands of tales spanning many different genres. Should one choose different versions of the same tale? Should one choose all "hero tales"? Finally, there is the problem of translation: more likely than not the language is different from the language for which our computational tools were designed. Can one expect to use translations and still obtain valid results?

Fortunately, these issues have already been dealt with. Propp selected a specific set of tales to analyze to derive his morphology. He took the first one hundred tales of a classic Russian folktale collection by Alexandr Afanas'ev (1957). While Propp did his work in the original language of the tales, Russian, for practical reasons I analyzed them in translation. Anthropologists have examined studying tales in translation, and the consensus is that, for structural analyses of the first order, the important semantic information of the tale comes across in the translation. "If one translated a tale into another language, the tale structure and the essential features of the tale images would remain the same..." (Fischer 1963, p.249).

Propp, in his Appendix III, provided function markings for about half of the tales he analyzed: in the English translation of Propp's work there are only 45 tales in the function table, with a small number of additional analyses distributed throughout the text. Because I restricted myself to single move tales, the set of possible candidates is further reduced; across several different translations of Propp, only 21 single-move tales with function analyses were provided. My ability to annotate this set was further reduced by both readily accessible high-quality translations and my annotation budget. In the end, I was left with fifteen single-move tales which I was able to fully annotate, for a total of 18,862 words. This is the data used for the work, and is covered in detail in Chapter 4.

Chapter 3

Analogical Story Merging

In this chapter I describe the basic ideas behind the Analogical Story Merging (ASM) algorithm, which enables the learning of Propp's morphology. In the previous chapter I described how my learning strategy was to focus on learning the function categories and regular grammar of Propp's intermediate level. Consequently, Analogical Story Merging is based upon *Bayesian model merging*, a machine learning technique for learning regular grammars (Stolcke and Omohundro 1994). Bayesian model merging is a technique for learning regular grammars from positive examples, and falls into the class of grammar induction techniques that start with a specific, non-generalized model and then generalize it to a more acceptable form.

There are two key augmentations of ASM over Bayesian model merging. First, Bayesian model merging assumes the alphabet is known, whereas when learning Propp's morphology a major challenge is to learn the function categories simultaneously with the grammar. To achieve this, ASM incorporates a *filtering* stage for identifying the alphabet from the final model. Second, ASM operates over extremely complex data that are represented at multiple levels, and each state in the model has its own internal structure relevant to generalization decisions. Therefore, ASM incorporates an *analogical mapper* which allows the algorithm to consider this information when deciding which generalizations to pursue.

This chapter discusses only the basic ideas behind ASM. The details of the specific implementation of the algorithm and its performance over the Russian folktale data are discussed in Chapter 5. To make the distinction clear, I will refer to the general learning technique as *ASM*, and the specific implementation described in Chapter 5 targeted at Propp's morphology as *ProppASM*.

Here I first review the details of Bayesian model merging, and then explain the filtering and analogical mapping modifications that transform Bayesian model merging into Analogical Story Merging. Next, I discuss three dimensions along which ASM can be customized. Finally, I outline some initial results that illustrate the feasibility of the technique and identify two search optimizations that are critical to achieving convergence on real data.

3.1 Bayesian Model Merging

Model merging (Omohundro 1992) is the conceptual foundation of Analogical Story Merging. Model merging may be used to derive a regular grammar from a set of positive examples. Consider the set of two characters sequences {`ab,abab`}. What is the pattern that explains these two sequences? One guess is the regular grammar `(ab|abab)`. Our intuition, however, is that this guess is unsatisfactory because it does not generalize beyond the examples provided. Anyone can see that a more plausible guess is `ab` repeated one or more times, or, written as a regular expression, `(ab)+`. Bayesian model merging is a technique that finds, given the sequences, a good approximation to this pattern by, as the name suggests, applying Bayes' rule to making decisions about how to search the space of possible grammars.

Model merging follows the grammar inference paradigm that starts with a model constructed to accept the finite language composed of exactly the positive examples observed (Young-Lai 2009). Generalization is achieved by applying a *merge operation* over states in the model where two states are removed from the model and replaced with a single state that inherits the removed states' transitions and emissions. This induces a large space of models to search. Because this is *Bayesian* model merging, one applies a given prior over models, and Bayes' rule, to evaluate the fitness of each possible model encountered and so guide the search. Moreover, I can apply many well known search strategies to improve the search's time or space performance metrics.

The technique is illustrated in Figure 3-1 and proceeds as follows. First, generate an initial HMM, called M_0 in the figure, by incorporating each example explicitly into the model. This HMM has six states, each of which emits a single character, either `a` or `b`, and generates with 50% probability either the sequence `ab` (top branch) or the sequence `abab` (bottom branch). Second, define a prior over models, which is a probability mass function for how probable a model is *a priori*. This example uses a geometric distribution, which says that smaller models are more likely. The model itself can be used to calculate the probability of seeing the observed data (it is 25% for M_0). Using Bayes' rule, one can calculate a number proportional to the posterior, which is the probability of the model given the observed data.

The search is driven by trying to the find the model that maximizes the posterior. In Figure 3-1, states shaded in one step are merged together into the dashed state in the next step. The figure shows the step-by-step progression from the initial model through a series of merges (found by search) to the final model which maximizes the posterior. The first merge combines states 1 and 3 generating model M_1. This model still only produces the original two examples, but it is smaller than M_0, so it is more probable. The second merge combines states 2 and 4 to produce M_2. Again, this does not change the output, but does produce a smaller model. The third merge combines states 2 and 6 to produce M_3, and in this step generalization occurs, in that the HMM can now produce any string that matches `(ab)+`. This reduces $P(D|M)$, the probability of the data given the model, but by not as much as is gained from the increase in the prior. The final merge produces M_4, a smaller model, and any further merging causes a reduction in the posterior.

The search space for Bayesian model merging is quite large, equal to Bell's number, B_n, where n is the number of initial states in the model. Bell's number counts the number of unique partitions of a set of n objects (Rota 1964), which grows dramatically with n. While there is no known closed form, it has been shown (Berend and Tassa 2010) to be closely bounded above by Equation 3.1.

$$B_n < \left(\frac{0.792n}{ln(n+1)} \right)^n \tag{3.1}$$

Table 3.1 illustrates how Bell's number grows with n; for search spaces of these sizes an exhaustive search is only possible for the smallest examples. Of the problems discussed, only the toy example (§3.4.1) falls into that range. Our first problem of any interest, the Shakespearean plot summaries (§3.4.2), already outstrips the number of stars in the observable universe by a considerable amount. Because of this fact, a large part of the strategy will involve techniques for pruning the search space, and heuristics to quickly find an approximately-best answer.

3.2 ASM Augmentations

Although model merging forms the foundation for Analogical Story Merging, there are two key differences: filtering and analogical mapping.

3.2.1 Filtering

I sought to learn not only the structure of the regular grammar of the morphology's function level, but also its "alphabet" — the functions themselves. To do this, I first began with an alphabet of all *possible* symbols. The key was to provide an overly-generous definition of the alphabet, so it was known that some proper subset of it represents the actual alphabet. The model merging search was performed as usual. Finally, once the search portion of the algorithm completed and

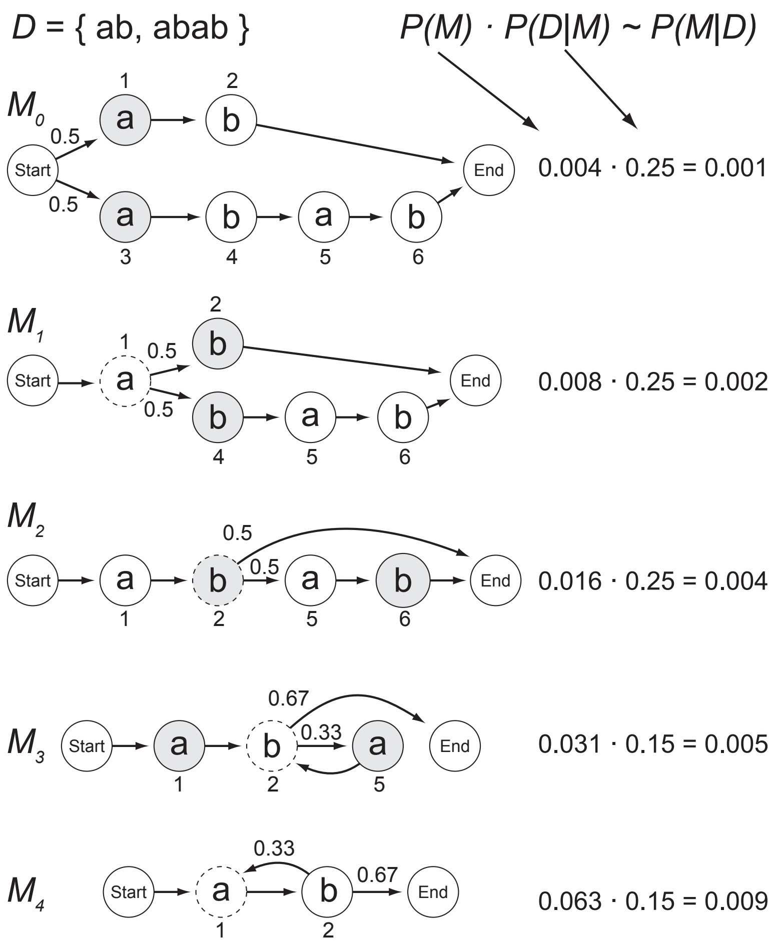


Figure 3-1: Example of Bayesian model merging, after (Stolcke and Omohundro 1994, Figure 1). The original character sequences are labeled D. States are represented by circles and transitions by arrows. The symbols (either `a` or `b`) emitted by a state are listed inside its circle. States are numbered, and these appear outside each state's circle. States shaded in one step are merged into the dashed state in the next step. The probabilities of the prior (geometric, $p = 0.5$) and the data given the model are given on the right, along with their product, which is proportional to the posterior probability. The merging proceeds until the product cannot be further increased.

the posterior-maximizing model (or the closest candidate) is found, that model was *filtered*. The filtering process constructs another model from the final merged model from which all states that do no meet certain criteria are removed. A well-engineered prior is critical to ensuring that the retained states correspond to the structures which are of interest. The states that survive this culling become the alphabet, or for Propp's morphology and ProppASM, the functions. This process is illustrated in Figure 3-2.

3.2.2 Analogical Mapping

The other difference between ASM and normal model merging is that the states in the model are not atomic. States in the model have internal structure which affects model fitness. For deeply semantic domains, this internal structure is critical to determining the correct model. In the general implementation of ASM, I incorporated the Structure Mapping Engine (Falkenhainer, Forbus, and Gentner 1989), a general structure alignment algorithm that is customizable with different sets of alignment rules. The mapper not only outputs a simple match score that can be used in the fitness

Relevant Problem	n	B_n
	2	5
	3	15
	4	52
	5	203
	6	877
	7	4,140
Toy Example (§3.4.1)	8	21,147
	9	115,975
	10	678,570
Shakespeare Example (§3.4.2)	42	5.53×10^{38}
	100	1.40×10^{117}
Plot Unit Example (§3.4.3)	103	3.83×10^{121}
	500	1.71×10^{845}
Filtered Propp Corpus (Chap. 5)	904	3.89×10^{1712}
	1,000	5.69×10^{1929}
Unfiltered Propp Corpus (Chap. 5)	2,263	3.11×10^{5024}

Table 3.1: Growth of Bell's number with n. The column on the left corresponds to problems described in the indicated section or chapter. Even for small initial models (42 initial states), the unpruned search space is unmanageably large.

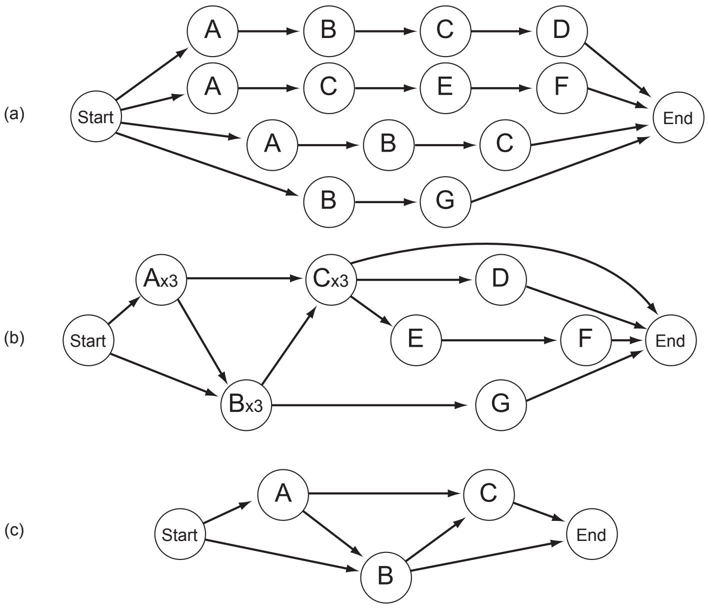


Figure 3-2: Example of filtering the final model of ASM to identify the alphabet of the model. (a) An initial ASM model, completely unmerged. State symbols A through G are extracted from the data and indicate potential symbols in the morphology. (b) The ASM algorithm searches the merge space and produces this final merged model, which has three highly merged nodes: A, B, and C. (c) Filtering out all unmerged nodes results in this model, which identifies both the structure and the alphabet of the grammar.

function (in the Bayesian case, the prior), but it also allows access to the details of a structural alignment so that more fine-grained judgements can inform the merge process.

3.3 ASM Parameters

Analogical Story Merging has three dimensions that can be varied to adjust the algorithm to different needs: how to construct an initial model from the data, what sorts of merges are allowed, and the form of the prior.

3.3.1 Initial Model Construction

For stories there is no straightforward mapping from the story texts to "strings" in the target language. Even access to the stories' formal representations, as discussed in the next chapter, does not reduce the number of options. There are a large variety of reasonable ways to extract a linear sequence of elements from stories for the purpose of building the ASM initial model. How one does it depends on the sort of narrative structure one is interested in extracting. Because Propp was interested in the events in the story and how they were instantiated by particular functions, the obvious choice for reproducing Propp's morphology is to use the events to construct a linear timeline for each story that can then be used to construct an initial model. On the other hand, if one were to desire to extract structures such as those described by Levi-Strauss (1955), one would need to start with a completely different set of basic elements.

But, even constrained to event timelines, there are still a number of ways to proceed. Oftentimes the order of the events as expressed in the text is different from their temporal order within the story world. A simple example is a construction of the form "He did A. But before that, he did B." A comes before B in the text, but in the timeline of the story, B occurs before A. A more complex example is a *meanwhile* construction, in which the story focuses on events in one location, and then later in the text switches to simultaneous events at another location.

Moreover, as I shall describe in more detail in Section 4.2.4, there are a number of different types of events that are expressed in stories. These include, to mention but a few, straightforward things that happen ("He fought the dragon."), times and dates ("He fought the dragon on Tuesday."), aspectual constructions ("He began to fight the dragon."), and subordinative expressions ("He promised he would fight the dragon, but never did."). Each of these types of occurrences has a particular meaning for the story as a whole and a particular place in the timeline of the story world. When constructing the linear sequence of events, one might exclude some story events *a priori*. Furthermore, one might include other items, where they are not expressed directly as an event, but imply an event or other relevant timeline entity. One might make a single state out of one or more sequential events.

In the examples presented, I derived the initial model from the timeline of events in the story world itself. For the examples presented, I defined each symbol to be a single event in the story and their order to be the order in which they occur in the story timeline. Each individual story timeline was then incorporated into the initial morphology as a single, linear branch. An example initial morphology, labeled M_0, can be seen at the top of Figure 3-5, where each of the two simple example stories with their four constituent events is transformed into a sequence of four states.

3.3.2 Merge Operations

How the algorithm merges two states into one when searching for a more compressed and generalized model is also subject to variation. In all the problems discussed, I use Stolcke & Omohundro's merge operation, where the merged state inherits the weighted sum of the transitions and emissions of its parents. The representations contained in each event are atomic for the purposes of merging, so each state can emit any event that is agglomerated into it. That is, if a state S_1 emits only event A, and state S_2 emits only event B, then for the purposes of calculating probability of the data given the model, a state created from merging S_1 and S_2 has a 50% chance of emitting either A or B.

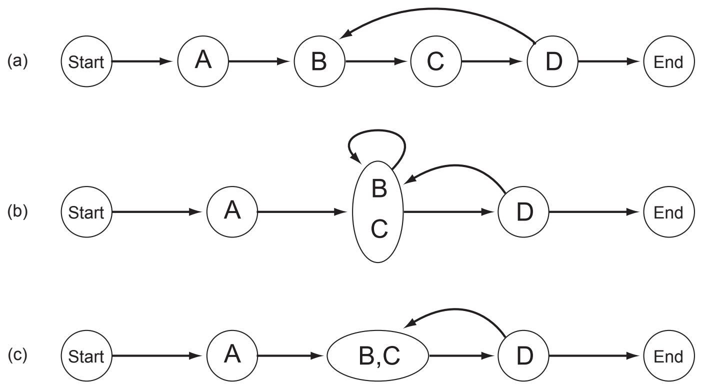


Figure 3-3: Example of a lateral merge operation. (a) A portion of a model to be merged. (b) A normal merge operation. Events B and C have been merged together, creating a state that emits either B or C each with probability 0.5. The new state inherits all transitions of the old B and C states, and so it has a transition to itself. (c) A lateral merge operation. Events B and C have been concatenated, so the merged state emits the event pair BC with probability 1. It also lacks a self-transition.

There are, however, other possible merge operations. These merge operations might augment or generalize the contents within the states. For example there is a type of merge I call the *lateral merge*, in which if one were merging two states that were from the same story, and they were adjacent, their internal structures would merely be concatenated rather than agglomerated. This is illustrated in Figure 3-3. Because lateral merges avoid creating some state-to-self-transitions, this could potentially create a better final model in cases where there are several expressions of an individual function in quick succession in a story.

3.3.3 Prior Distribution

In the cases where one uses a Bayesian fitness function to guide the search, there are several important degrees of freedom, all having to do with the form of the prior.

First, the prior should take into account the overall structure of the model. This can be as simple as a uniform prior over all models or a prior which considers only the number of states in the model, both of which are common in Bayesian structural discovery techniques (Tenenbaum et al. 2011). Often, with no information to the contrary, the simplest approach is to assume morphologies with fewer states should have higher probabilities. This can be expressed by a simple function that either decreases monotonically with the number of states (e.g., the geometric distribution) or perhaps simply peaks at the *a priori* most likely number of states (e.g., the beta distribution with appropriate parameters). More complex priors, such as the Chinese Restaurant Process (Pitman 2006), can take into account not only the number of states, but also how highly merged they are and the overall distribution of merges across states.

Second, the prior should modulate the probability according to the internals of each state, in that states with similar events should be more probable than states with dissimilar events. For example, if event A is similar to B, but not similar to C, then a state S_1 containing events A and B, should have a higher prior probability than another state S_2 containing events A and C. This is illustrated in Figure 3-4. Via this approach the search is biased toward models that group similar events together into the same state. The simplest implementation of this rule is to disallow non-similar events in the same state, which allows for an efficient search because those portions of the search space are zeroed out.

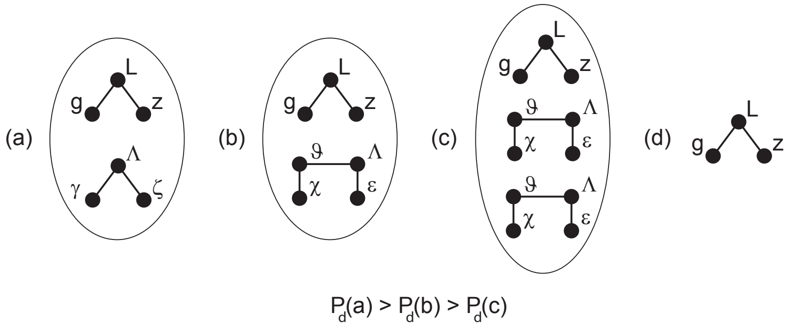


Figure 3-4: Examples modulating conditional probability on the basis of state internals. (a) A state containing two similar events with similar structures and semantics. (b) A state with two dissimilar structures. (c) A state with an additional dissimilar structure. (d) Because we would like similar events to be merged together, the prior probability of each state emitting the g-L-z structure should obey $P(a) > P(b) > P(c)$

3.4 Examples

I present three examples of using Analogical Story Merging to extract morphology-like structures.

3.4.1 Toy Example

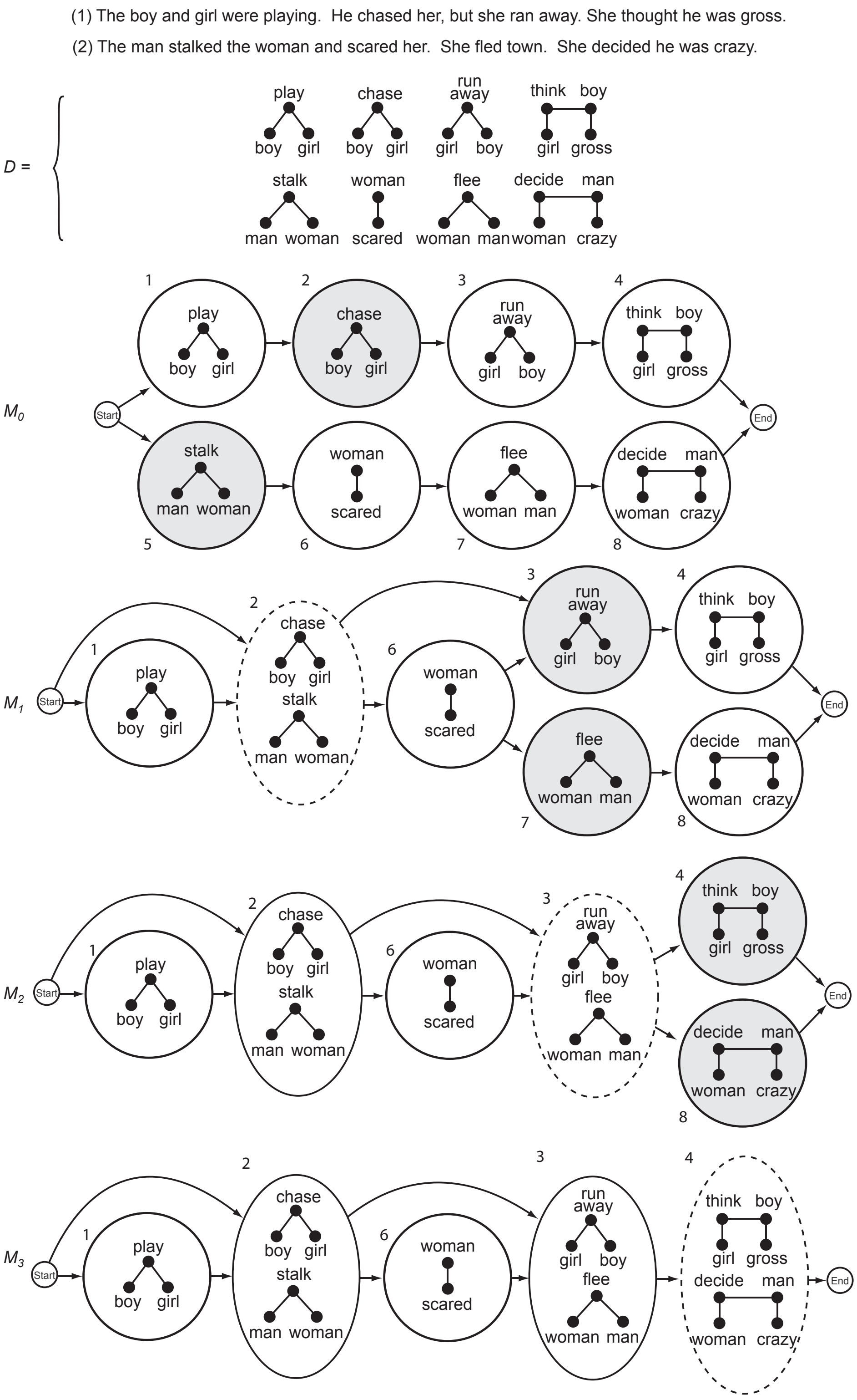
Figure 3-5 illustrates Analogical Story Merging by showing the extraction of a simple morphology from two extremely short stories. The first story has to do with a boy and girl playing, a chasing event, a running away event, and ending with a thinking event. The second story has to do with a man stalking a woman, followed by a scaring event, a fleeing event, and ending with a decision event. At some level of analysis these two stories are similar. The chasing and stalking events are similar in that they involve one participant following after another, the running away and fleeing events are similar because they involve movement of one participant away from the other, and the thinking and deciding events are both mental events that involve evaluation. If one represents these aspects of the semantics of these events, or some equivalent measure, one can use an analogical mapping algorithm to find the semantic and structural similarities. In the set of merges shown, first the chasing and stalking events are merged, then the running away and fleeing, and then the thinking and deciding events. This results in a story morphology that generates stories with an optional playing event at the beginning, a pursuit event, followed by an optional scared event, followed by the fleeing and evaluation events. Once the final model is filtered three states remain, which may be titled *Pursuit*, *Flee*, and *Judgement*.

For this example, the prior is a geometric distribution with parameter $p = 0.95$, multiplied by the product of the individual probabilities of each state in the morphology, where the probability of each state is 1 if all the events in the state are pairwise similar, and 0 otherwise:

$$P(M) = p(1-p)^{n-1} \prod_i K(S_i) \tag{3.2}$$

$$K(S_i) = \begin{cases} 1 & \text{if } \forall e_j, e_k \in S_i, \quad Sim(e_j, e_k) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

In Equation 3.2, M is the model, p is the single parameter of the geometric distribution, n is the number of states in the morphology, and S_i is the ith state. In Equation 3.3, e_j and e_k are events in state S_i, and Sim is the similarity function. For these examples, the similarity function is implemented by the Structure Mapping Engine, where two events are considered similar if they have a complete mapping (i.e., every item in the event is mapped to some item in the other event),



(1) The boy and girl were playing. He chased her, but she ran away. She thought he was gross.

(2) The man stalked the woman and scared her. She fled town. She decided he was crazy.

Figure 3-5: Example of Analogical Story Merging on two simple stories. Shown is a series of merges leading to the model that maximizes the posterior under the parameters of ASM described. The final model describes not only the two input stories, but an additional two stories that alternatively include or exclude both nodes 1 and 6. Thus the model has generalized beyond the two input examples.

and dissimilar otherwise.[1]

3.4.2 Shakespearean Plays

In this example I extract narrative similarities from a small story corpus comprising summaries of five plays by Shakespeare, namely, *Macbeth*, *Hamlet*, *Julius Caesar*, *Othello*, and *Taming of the Shrew*. The summaries were originally written in simple controlled English for another analogy system (Winston 1980a); the original controlled English is given in (Winston 1980b). Each summary contains between 7 and 11 events, totalling 43 events. The prior is the same as in Equation 3.2. The final, unfiltered model is shown in Figure 3-6.

The final model is interesting for at least three reasons. First, it captures important plot similarities and differences, indicating where one plot branches off from another, and where they merge together again. The paths through the model representing the four tragedies share states, while the path representing the single comedy, *Taming of the Shrew*, is in its own separate branch. *Hamlet* and *Macbeth* share a main branch {2,5,6}, with *Hamlet* detailing *persuasion* leading to the murder, and *Macbeth* adding a detour through the sequence {7,8,9,10} between states 2 and 5. They are more similar to each other than to *Julius Caesar*, which shares only the initial persuade-murder sequence. These similarities are consistent with analogical mapping considerations, in that a pairwise mapper comes to the same similarity conclusions (Winston 1980a). Also, events that are similarly key are grouped together. *Macbeth*, *Hamlet*, and *Julius Caesar* all have murders as motivating events; these are grouped into states 4 and 7. Both *Hamlet* and *Macbeth* conclude with a “revenge achieved” event, involving the killing of the murderer(s); these are grouped together into state 6.

Second, the model generalizes from the stories presented. This generalization is especially evident at states 9, 10, 20, 22, and 24, where multiple events occurring in sequence have been collapsed to a single state that allows an arbitrary number of events of those type to occur. State 10 contains the event where the Ghost orders Hamlet to kill Claudius, and might be thought of as a generalized “conspiracy” event. States 12 and 13 contain killing or attacking events close in time, and each can be thought of as a “fight.”

Third, the model captures when something unusual happens, and sets it off by itself. This is most evident with the whole of the *Taming of the Shrew*, which is sufficiently dissimilar from any other play in the set that it is relegated to its own branch. This also covers state 1 , the event in *Macbeth* involving the three witches predicting things to come; despite strong starting similarities between *Macbeth*, *Hamlet*, and *Julius Caesar*, there is no analogous event, semantically, structurally, or temporally, in any other story in the set, and so the sisters are set off by themselves.

There are a number of peculiarities to note. First, it is curious that states 4 and 7 have not been merged. Their constituent events are certainly similar, and the events in question occur one after another. A similar sequence of four *attack* events are merged into the single state 12. So why not merge 4 and 7? The only story in which multiple murdering events occur in sequence is *Julius Caesar*[2], and thus merging those states together decreases $P(D|M)$ more than the increase in prior gained from reducing the model size by one state. Second, it is curious that states 5 and 13 were not merged, which would have agglomerated all the suicides that occur at the ends of the stories. However, merging the two would make it more likely that a Julius-Caesar-like story would have an additional killing at the end (rather than a suicide), and this movement of probability mass away from the original data was not offset by the reduction in model size. In general, all questions of “to merge or not to merge” reduce to this tradeoff between compacting the model and losing fit with the original data.

[1] Equation 3.2 does not define an actual probability, in that it does not sum to one over all possible models. This is of no concern because we are merely interested in finding the morphology that maximizes the posterior, not the posterior’s actual value.

[2] Caesar’s murder is represented as two events, first a murder by Brutus, then a murder by Cassius.

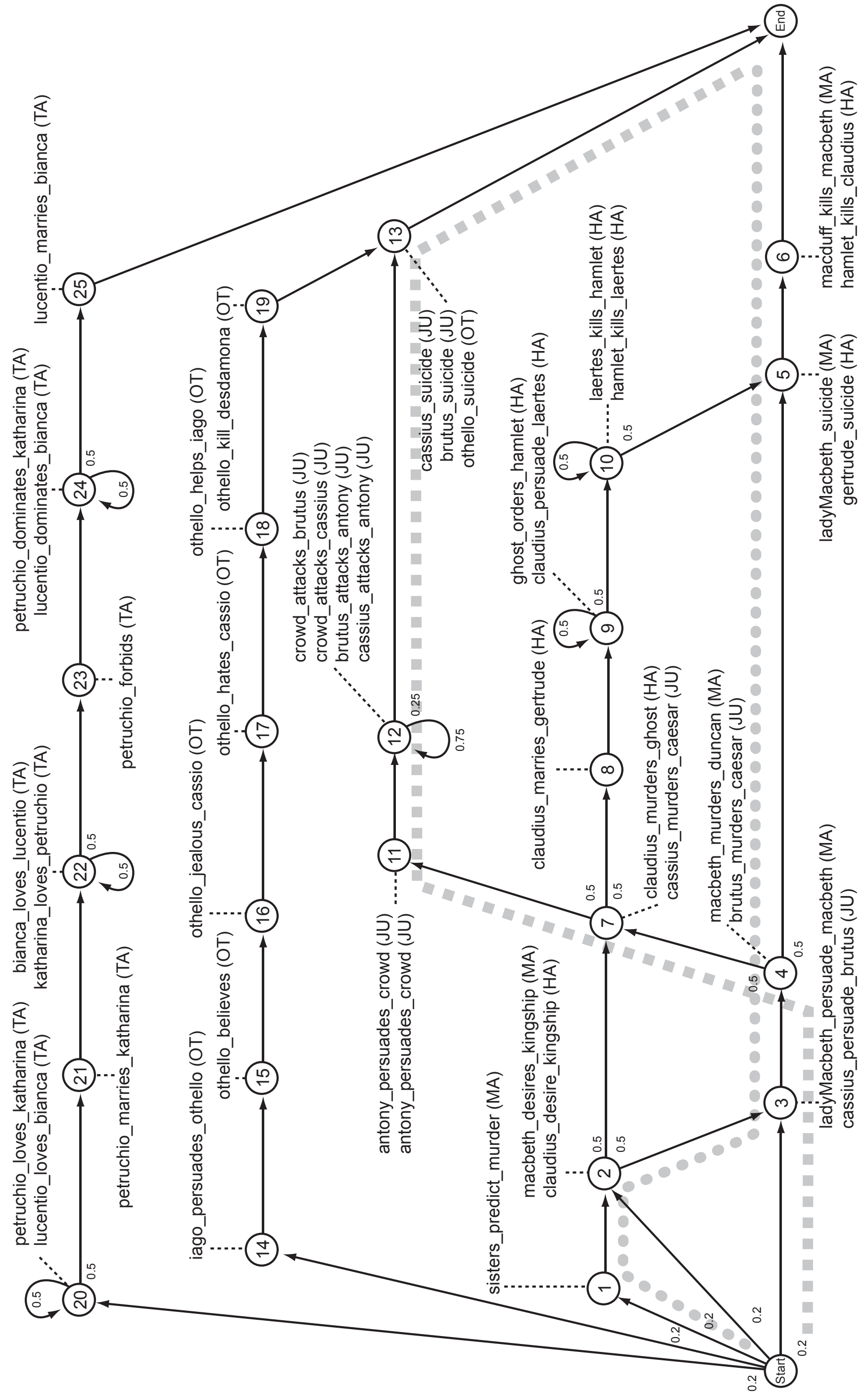
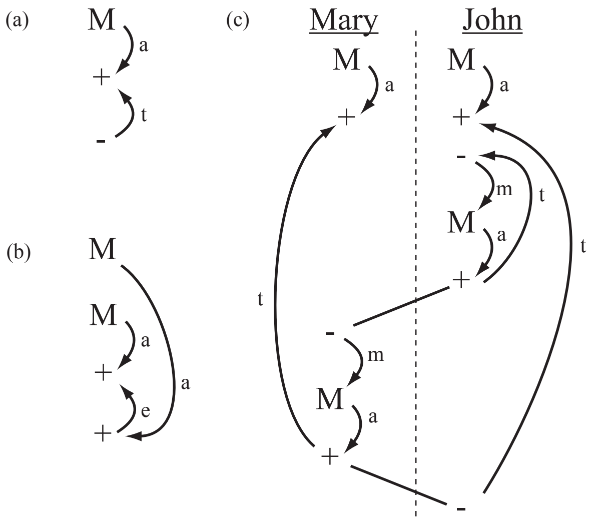

Start
End
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
0.2
0.5
0.25
0.75
sisters_predict_murder (MA)
macbeth_desires_kingship (MA)
claudius_desire_kingship (HA)
ladyMacbeth_persuade_macbeth (MA)
cassius_persuade_brutus (JU)
macbeth_murders_duncan (MA)
brutus_murders_caesar (JU)
claudius_murders_ghost (HA)
cassius_murders_caesar (JU)
claudius_marries_gertrude (HA)
ghost_orders_hamlet (HA)
claudius_persuade_laertes (HA)
laertes_kills_hamlet (HA)
hamlet_kills_laertes (HA)
ladyMacbeth_suicide (MA)
gertrude_suicide (HA)
macduff_kills_macbeth (MA)
hamlet_kills_claudius (HA)
antony_persuades_crowd (JU)
antony_persuades_crowd (JU)
crowd_attacks_brutus (JU)
crowd_attacks_cassius (JU)
brutus_attacks_antony (JU)
cassius_attacks_antony (JU)
cassius_suicide (JU)
brutus_suicide (JU)
othello_suicide (OT)
iago_persuades_othello (OT)
othello_believes (OT)
othello_jealous_cassio (OT)
othello_hates_cassio (OT)
othello_helps_iago (OT)
othello_kill_desdamona (OT)
petruchio_loves_katharina (TA)
lucentio_loves_bianca (TA)
petruchio_marries_katharina (TA)
bianca_loves_lucentio (TA)
katharina_loves_petruchio (TA)
petruchio_forbids (TA)
petruchio_dominates_katharina (TA)
lucentio_dominates_bianca (TA)
lucentio_marries_bianca (TA)

Figure 3-6: The morphology produced by running Analogical Story Merging over a set of five summaries of Shakespearean plays: *Macbeth*, *Hamlet*, *Julius Caesar*, *Othello*, and *Taming of the Shrew*. Each event is marked with the first two letters of the play from which it came. As an example, the shaded dashed lines indicate the path of *Macbeth* (circle dashes), *Julius Caesar* (square dashes), and their overlap, which occurs in states 3 and 4. *Macbeth* and *Hamlet* have the most plot overlap, as expected, followed by *Julius Caesar* against *Macbeth* and *Hamlet* (Winston 1980). *Othello* shares the final suicide with *Julius Caesar*. The *Taming of the Shrew* is the only comedy, and is set off in its own completely separate track.



John was thrilled when Mary accepted his engagement ring. But when he found out about her father's illegal mail-order business, he felt torn between his love for Mary and his responsibility as a policeman. When John arrested the old man, Mary was furious and called off the engagement.

Figure 3-7: Examples of complex plot units. (a) A *Fleeting Success* starts with a goal (M) that results (a) in the goal's accomplishment ($+$) which is followed by the goal returning (t) to an accomplished state ($-$). (b) *Killing Two Birds* starts with two goals (M and M), one of which results (a) in accomplishment ($+$). That accomplishment is then equivalent (e) to the accomplishment of the other goal (a). (c) A complete parse of the short story shown in the box into elements and connections, after (Lehnert 1981, Figure 26).

3.4.3 Lehnert-style Plot Units

The third and final example[3] concerns extracting Lehnert-style plot units from examples. Lehnert's theory of plots units (Lehnert 1981) was important early work in representing the higher-level structure of stories. Her theory shares important features with Propp's morphology, and, in particular, plot units are rough analogues of Propp's functions. Like Propp's morphology, one main open question with Lehnert's work is the provenance of the plot units. In her seminal papers on the topic, she gives an exhaustive list of what she calls *primitive* plot units, and numerous examples of *complex* plot units. Her list of complex plot units, however, is not exhaustive, and she speculates that the set of available plot units will vary depending on the individual, the context, or the culture. Here we show that ASM can extract plot unit structures from stories. In particular, when given four example stories — two with an embedded with a *Revenge* plot unit, and two with embedded with a *Pyrrhic Victory* plot unit — ASM can find both patterns. The example is important because it shows that ASM works as expected on simple inputs. It also introduces an important heuristic that significantly prunes the search space, namely, character mappings.

A plot unit is built out of three elements and five connections, as illustrated in Figure 3-7. The elements represent three types elements: mental states (M), and positive ($+$) or negative ($-$) affect states. A time-ordered list of these elements is produced for each character in a story. Elements within a single character's timeline can be connected by one of four causal connections: Motivation (m), actualization (a), termination (t), or equivalence (e). Affect states between character timelines are connected by cross-character causal links (diagonal lines) indicating that those states are reactions to the same event in the world.

[3] This section describes joint work with my student, Brett van Zuiden.

Revenge 1: In early 2010, Google's servers were attacked by Chinese hackers. As such, Google decided to withdraw from China, removing its censored search site and publically criticizing the Chinese policy of censorship. In response, a week later China banned all of Google's search sites.

Revenge 2: In 1998, Afghan terrorists bombed the U.S.'s embassy in Cairo, killing over 200 people and 12 Americans. Two weeks later, The U.S. retaliated for the bombing with cruise missile attacks on the terrorist's camps in Afghanistan, which were largely unsuccessful. The terrorists claimed that the bombing was a response to America torturing Egyptian terrorists several months earlier.

Pyrrhic Victory 1: In 2002, as part of its program to maintain strict control over information, China blocked Google entirely. Although they later unblocked it, Google wanted to prevent such an occurrence in the future, and so in 2005 made a compromise with China: Google would filter its search site if China allowed Google to operate in China. China agreed, but the move caused many to criticize Google for cooperating with China's overbearing censorship policies.

Pyrrhic Victory 2: Over the last 10 years, Apple has been trying to increase its market share and popularity, and has succeeded in doing so, with Macs now comprising two-thirds of the high-end computer market. This increased popularity, however, has also led to increased numbers of malware attacks on Apple's computers, and Apple now recommends using antivirus software.

Figure 3-8: Pyrrhic Victory and Revenge stories used in the plot unit learning experiment. Note that these stories are not necessarily historically accurate; they were constructed for the purpose of the experiment.

Story	Type	Explicit	Inferred	Total
Apple's Market Share	Pyrrhic Victory	7	8	15
China blocks Google	Pyrrhic Victory	12	5	17
Hackers attack Google	Revenge	11	25	36
Cairo Embassy Bombed	Revenge	16	9	35
Total				103

Table 3.2: Event counts for stories used to learn the Revenge and Pyrrhic Victory plot units.

Lehnert gives numerous examples of complex plot units in her work. Some examples are shown in Figure 3-7. For a given story one can write down a graph of affect states and connections that describes the interactions in the story; plot units are then found as patterns in this graph. The figure also shows a story text along with its corresponding manually constructed affect state parse.

In this example, we used ASM to extract two simple plot units, *Revenge* and *Pyrrhic Victory*, from four story texts. Two additional stories were used for testing. These plot units were defined as indicated in the figure. We generated six short stories, three of which contained *Pyrrhic Victory* and three *Revenge*. Then four stories were used to learn the plot units (listed in Figure 3-8), two from each type.

These stories were annotated (as described in Chapter 4) using the Story Workbench. Mental state and causal connections, key elements in Lehnert's representation, are not part of the Story Workbench's suite of representations. These features are usually implicit in stories, being provided by the user's commonsense understanding of the world. To provide this commonsense knowledge, we processed the Story Workbench representations with the Genesis Story Understanding system (Winston 2011). Using an if-then-rule forward-chaining commonsense reasoner, the Genesis system augmented the story representations with mental state and causality information. These augmented representations were then provided to the ASM algorithm. A simple geometric prior was used with a p-value parameter of 0.95.

Importantly, we developed a search heuristic we call *character maps* for this example. This search heuristic significantly prunes the search space for complex stories. The insight behind this

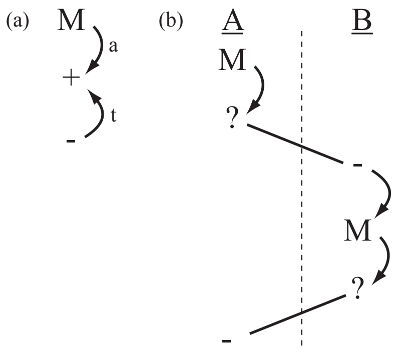


Figure 3-9: Complex plot units Revenge and Pyrrhic Victory. (a) Revenge is when one person causes harm to another, and then is harmed in return. (b) Pyrrhic Victory is when you achieve a goal of your own, but that goal is also a defeat.

heuristic is that if one proposes a merge that implies person A from story 1 is equivalent to person B from story 2, you should not propose another merge in which person A is equivalent to someone other than person B. In this implementation, the system first decides on a character map between the stories, and then only allows merges consistent with that map. For the actual search algorithm, beam search with a width of 40 sufficed to converge to the correct answer on most runs.

As described in Section 3.2, the highly merged nodes of the final graph were filtered to find the plot units. Because a great deal of repetitive information was inserted into the graphs in the form of commonsense knowledge, any patterns that matched a commonsense knowledge rule were also removed. This allowed ASM to extract the Revenge and Pyrrhic Victory patterns, shown in Figure 3-9, which could then be matched to additional test stories.

Chapter 4

Corpus Annotation

I turn to the data from which Propp's morphology was to be learnt. The data were English translations of Afanas'ev's folktales, annotated for various aspects of the text's syntax and semantics. I first describe the basics of annotation, the process of attaching additional information to text to make aspects of its meaning explicit. Next I describe the suite of representations, or annotation types, that were annotated on the texts. Third, I analyze the quality of these annotations through a variety of metrics. Finally, I describe the details of the Story Workbench, a general text annotation tool that I developed in the course of this work which was used for all annotation.

4.1 Text Annotation

My use of the word *annotation* is the same as in corpus linguistics in that it covers "any descriptive or analytic notations applied to raw language data" (Bird and Liberman 2001). By *semantically annotated*, I mean the following: Given a natural language text, which is a specific sequence of characters including all symbols and whitespace, a semantic annotation will be the assignment to a subset of those characters another string of characters that conforms to a defined format. The format of an annotation will be called the *representation*, and the string attached to the text will be called the *annotation* or the *description*.

The semantic annotation of a text requires (1) the definition of a set of representations (the aspects of meaning we are interested in annotating) along with (2) the creation of a set of descriptions in those representations that are attached to specific points in the text (the pieces of meaning). Take the text in example (1).

(1) `John kissed Mary.`

There are 17 characters including two spaces and one period. I refer to subsets of characters by their indices, starting at 0. The first letter, *J*, spans indices [0,1], and the first word, *John*, spans indices [0,4], where `[x,y]` has the natural interpretation, namely, the span of characters in the text that starts at index `x` and ends at index `y`. Many different aspects of this text that may be made explicit. Each open class word could be annotated with the appropriate definition from the Wordnet electronic dictionary, using the Wordnet sense key (a lemma separated from a sequence of numbers by a percent sign) to identify the definition (Fellbaum 1998). Thus the string indicated in (2) might be attached to the word *kissed* spanning `[5,11]`.

(2) `kiss%2:35:00::`

Each sentence in the text might be assigned a Penn-Treebank-style syntactic analysis (Marcus, Marcinkiewicz, and Santorini 1993). Thus we might attach the string in (3) to the text span `[0,17]`, which indicates the syntactic structure of the sentence.

(3) `(S (NP (NN John)) (VP (VBD kissed) (NN Mary)) (. .))`

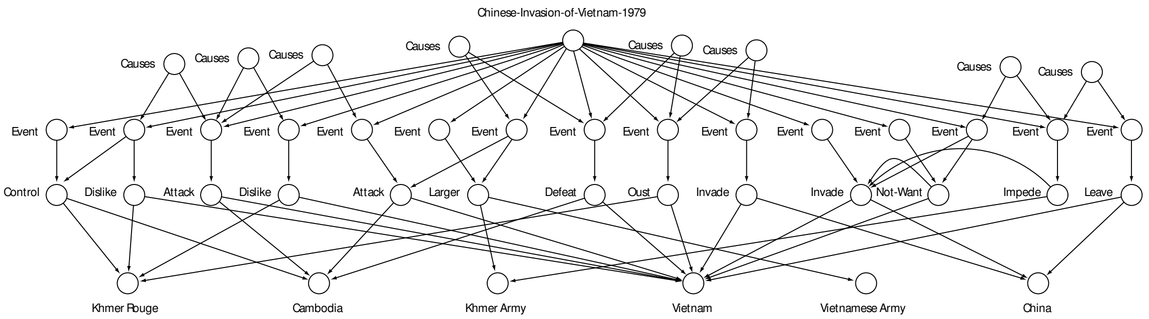


Figure 4-1: Graphical interpretation of a semantic annotation of synopsis of the Chinese-Vietnamese War of 1979. Events are ordered in temporal sequence from left to right. Taken from (Finlayson and Winston 2006).

It is tempting to think that annotations could be assigned to subsets of *words* rather than subsets of characters. This, however, is problematic because words themselves are annotations and could potentially be ambiguous. Is *don't* one word or two? How about the proper noun *New York City*? Character subsets are unambiguous and thus form the foundation on which all other annotations are built.

To learn Propp's morphology requires information on the timeline of events in the stories. Consider a set of annotations which represents events, their order, and their participants. Figure 4-1 shows a graphical interpretation of such a set of annotations for a synopsis of the 1979 Chinese invasion of Vietnam. Annotations such as these can be instantiated in a set of strings much like the syntactic parse tree above using three different representations, one to list the different participants and where they appear in the text, one to list the events and who is participating, and one to list the causal relationships between the events. From the size and complexity of this graph, which was produced from a mere 13 sentences of controlled English using a rule-based semantic parser, one can begin to appreciate the scope of the annotation problem when each sentence is of normal complexity, each text is hundreds of sentences long, and the size of corpus runs into the tens of thousands of words. Annotating such a corpus, and doing it well, is daunting to say the least.

4.1.1 Automatic Annotation

Automatically extracting meanings such as those illustrated in Figure 4-1 with high quality is beyond the reach of the current state of the art natural language processing technologies. Happily, the automatic extraction of certain specialized aspects of meaning are significantly advanced. For example, texts can be automatically split into tokens nearly perfectly. Assigning a part of speech to each token in a text can be done with extremely high accuracy by a statistical part of speech tagger (Toutanova et al. 2003). Similarly, statistical parsers can provide syntactic analyses of sentences at a lesser, but still good, accuracy (Klein and Manning 2003). And recently, there has been encouraging work on the use of statistically-trained classifiers to assign word senses to words (Agirre and Edmonds 2007) and identify and assign arguments to verbs (Pradhan et al. 2005).

The advantages of automatic techniques are that they are fast and consistent. They also excel at producing well-formatted results (even if the results are not *correct*). Unfortunately, taken as a whole, automated techniques lack coverage, are prone to significant error, and cannot produce all the sorts of annotations needed. For example, while there are wide-coverage, high-accuracy syntactic parsers, the available logical-form parsers have poor coverage and even worse accuracy.

4.1.2 Manual Annotation

Manual annotation, by contrast, relies on a human annotator's own natural understanding to annotate a text. The benefit is that humans can create annotations that we cannot yet create automatically. But human annotators have numerous problems. They are expensive. Training can be time-consuming and complicated. They must understand the representations they will be annotating and the often complex constraints those representations obey. They must translate their understanding into the formal structures needed by the computer, a task at which people are notoriously poor. Take, for example, generating a syntactic analysis in the Penn-Treebank format considered above. Now imagine using your favorite text editor to write down, a syntactic analysis in that format for this whole page of text. It would, no doubt, take a long time and the answers, compared across multiple annotators, would almost certainly not match well.

4.1.3 Semi-Automatic Annotation: The Story Workbench

The Story Workbench is a tool that facilitates the collection of semantic annotations of texts. It uses off-the-shelf natural language processing technologies to make a best guess as to the annotations, presenting that guess (if necessary) to the human annotator for approval, correction, and elaboration. This is neither fully manual, nor fully automatic, but rather *semi-automatic* annotation. The tool is similar in appearance to a word processing program: it has an editor area where one can enter and modify text, menus and buttons for performing operations on that text, and a variety of views showing supplementary information. Figure 4-2 shows a screenshot.

The Story Workbench's semi-automatic approach combines the best features of manual and automatic annotation. It uses automatic techniques where they are available. If the techniques have high-accuracy, these can be used with little-to-no supervision. Otherwise, their results can be presented to the human annotator for correction. Elsewhere, automation is used to tackle the tasks that are hard for people but easy for computers, such as checking long lists of formatting constraints, or generating possible answers by searching large solution spaces. Where the computer has absolutely no purchase on a problem, the human can take over completely, using special editors to create the annotations.

Central to this approach is a sophisticated graphical user interface that gives annotators the right information at the right times and does so in an intuitive and clear manner. Careful engineering was required to get the right "impedance match" between annotators and the computer. Many users of the modern computer are familiar with the devices that constitute a user-friendly graphical interface: a wide range of graphical modalities for conveying information, such as color, highlighting, movement, shape, and size; evocative icons; automatic correction of formats and well-known errors; quick feedback loops that show errors and warnings as they occur; wizards and dialogs that guide and constrain repetitive or formulaic tasks; solutions to common problems encapsulated in their own functions and offered for execution at the appropriate times.

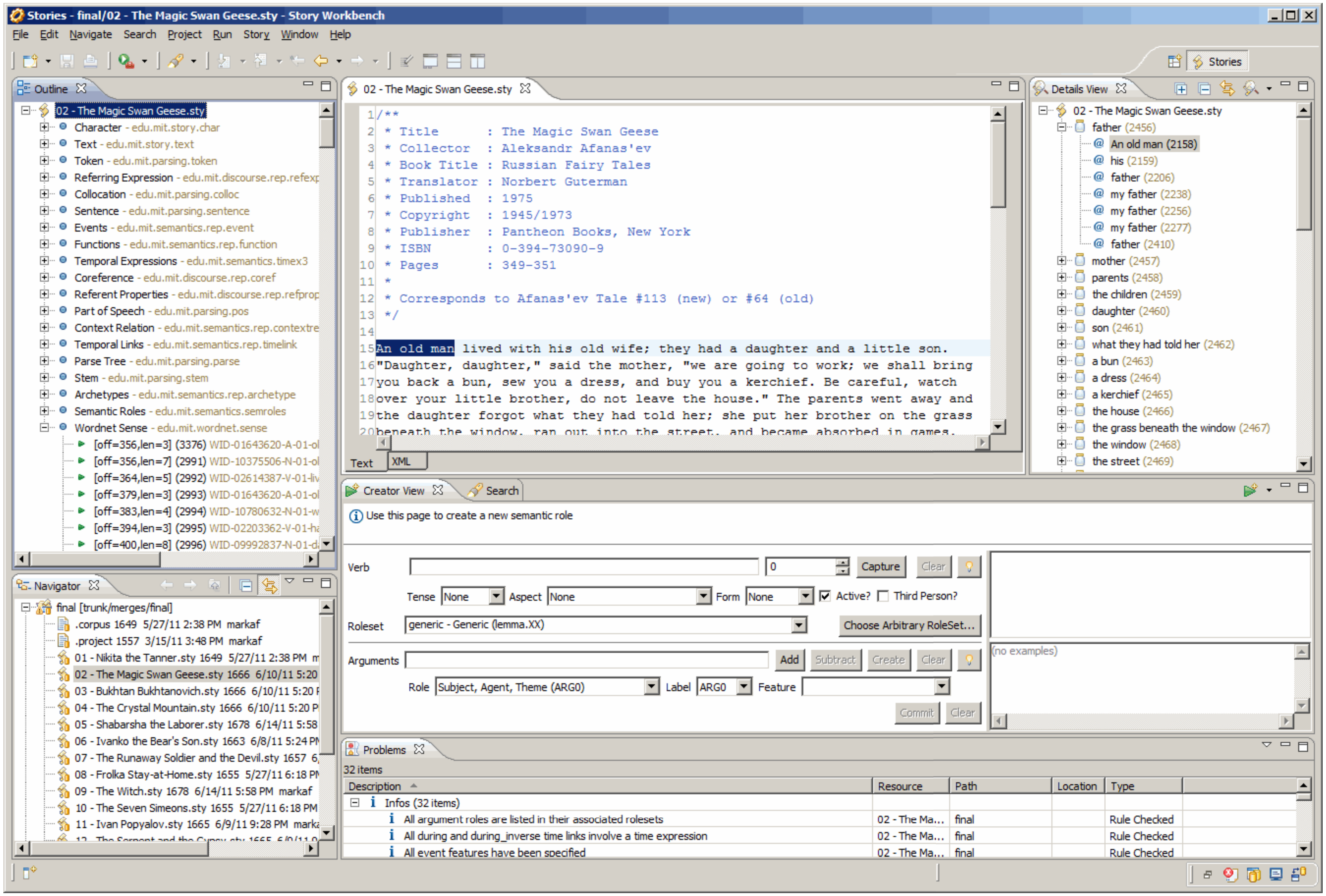
4.2 Representations

The Story Workbench is a general text annotation tool that supports arbitrary text annotations. Nevertheless, to fully harness its power one must carefully select and implement a suite of representations that will capture the meanings needed from the texts in the corpus. Table 4.1 lists the representations used in this work.

I explain how I processed these annotations to make the initial model for ASM is explained in the next chapter, Section 5.1. Here I explain each representation with examples, and describe how they were annotated.

4.2.1 Syntax

Syntax representations are the scaffolding on which the semantic representations are built. For my purposes, these representations are not interesting *per se*, but rather are mainly useful for calculating


Stories - final/02 - The Magic Swan Geese.sty - Story Workbench
File Edit Navigate Search Project Run Story Window Help
Stories
Outline
02 - The Magic Swan Geese.sty
Details View
Creator View
Search
Navigator
Problems
Text
XML

Outline

- 02 - The Magic Swan Geese.sty
  - Character - edu.mit.story.char
  - Text - edu.mit.story.text
  - Token - edu.mit.parsing.token
  - Referring Expression - edu.mit.discourse.rep.refexp
  - Collocation - edu.mit.parsing.colloc
  - Sentence - edu.mit.parsing.sentence
  - Events - edu.mit.semantics.rep.event
  - Functions - edu.mit.semantics.rep.function
  - Temporal Expressions - edu.mit.semantics.timex3
  - Coreference - edu.mit.discourse.rep.coref
  - Referent Properties - edu.mit.discourse.rep.refprop
  - Part of Speech - edu.mit.parsing.pos
  - Context Relation - edu.mit.semantics.rep.contextre
  - Temporal Links - edu.mit.semantics.rep.timelink
  - Parse Tree - edu.mit.parsing.parse
  - Stem - edu.mit.parsing.stem
  - Archetypes - edu.mit.semantics.rep.archetype
  - Semantic Roles - edu.mit.semantics.semroles
  - Wordnet Sense - edu.mit.wordnet.sense
    - [off=356,len=3] (3376) WID-01643620-A-01-ol
    - [off=356,len=7] (2991) WID-10375506-N-01-ol
    - [off=364,len=5] (2992) WID-02614387-V-01-liv
    - [off=379,len=3] (2993) WID-01643620-A-01-ol
    - [off=383,len=4] (2994) WID-10780632-N-01-w
    - [off=394,len=3] (2995) WID-02203362-V-01-ha
    - [off=400,len=8] (2996) WID-09992837-N-01-d

Navigator

- final [trunk/merges/final]
  - .corpus 1649 5/27/11 2:38 PM markaf
  - .project 1557 3/15/11 3:48 PM markaf
  - 01 - Nikita the Tanner.sty 1649 5/27/11 2:38 PM m
  - 02 - The Magic Swan Geese.sty 1666 6/10/11 5:20
  - 03 - Bukhtan Bukhtanovich.sty 1666 6/10/11 5:20 P
  - 04 - The Crystal Mountain.sty 1666 6/10/11 5:20 P
  - 05 - Shabarsha the Laborer.sty 1678 6/14/11 5:58
  - 06 - Ivanko the Bear's Son.sty 1663 6/8/11 5:24 PM
  - 07 - The Runaway Soldier and the Devil.sty 1657 6,
  - 08 - Frolka Stay-at-Home.sty 1655 5/27/11 6:18 PM
  - 09 - The Witch.sty 1678 6/14/11 5:58 PM markaf
  - 10 - The Seven Simeons.sty 1655 5/27/11 6:18 PM
  - 11 - Ivan Popyalov.sty 1665 6/9/11 9:28 PM marka

02 - The Magic Swan Geese.sty

```
/**
 * Title       : The Magic Swan Geese
 * Collector   : Aleksandr Afanas'ev
 * Book Title  : Russian Fairy Tales
 * Translator  : Norbert Guterman
 * Published   : 1975
 * Copyright   : 1945/1973
 * Publisher   : Pantheon Books, New York
 * ISBN        : 0-394-73090-9
 * Pages       : 349-351
 *
 * Corresponds to Afanas'ev Tale #113 (new) or #64 (old)
 */

An old man lived with his old wife; they had a daughter and a little son.
"Daughter, daughter," said the mother, "we are going to work; we shall bring
you back a bun, sew you a dress, and buy you a kerchief. Be careful, watch
over your little brother, do not leave the house." The parents went away and
the daughter forgot what they had told her; she put her brother on the grass
beneath the window, ran out into the street, and became absorbed in games.
```

Details View

- 02 - The Magic Swan Geese.sty
  - father (2456)
    - @ An old man (2158)
    - @ his (2159)
    - @ father (2206)
    - @ my father (2238)
    - @ my father (2256)
    - @ my father (2277)
    - @ father (2410)
  - mother (2457)
  - parents (2458)
  - the children (2459)
  - daughter (2460)
  - son (2461)
  - what they had told her (2462)
  - a bun (2463)
  - a dress (2464)
  - a kerchief (2465)
  - the house (2466)
  - the grass beneath the window (2467)
  - the window (2468)
  - the street (2469)

Creator View

Use this page to create a new semantic role

Verb: 0 [Capture] [Clear]

Tense: None Aspect: None Form: None ☑ Active? ☐ Third Person?

Roleset: generic - Generic (lemma.XX) [Choose Arbitrary RoleSet...]

Arguments: [Add] [Subtract] [Create] [Clear]

Role: Subject, Agent, Theme (ARG0) Label: ARG0 Feature:

[Commit] [Clear]

(no examples)

Problems

32 items

Description	Resource	Path	Location	Type
Infos (32 items)				
All argument roles are listed in their associated rolesets	02 - The Ma...	final		Rule Checked
All during and during_inverse time links involve a time expression	02 - The Ma...	final		Rule Checked
All event features have been specified	02 - The Ma...	final		Rule Checked

Figure 4-2: Screenshot of the Story Workbench main window, including the editor (upper middle), an outline of the file's contents (upper left), an outline of the corpus's files (lower left), the creator view (center middle), the problems list (bottom middle), and details view (upper right).

Representation	Description	New?	Annotation Style
Tokens	constituent characters of each token		Automatic
Multi-word Expressions	words with multiple tokens		Manual
Sentences	constituent tokens of each sentence		Automatic
Part of Speech Tags	tag for each token		Semi-Automatic
Lemmas	root form for each inflected word		Semi-Automatic
Word Senses	dictionary sense for each word		Manual
CFG Parse	grammar analysis of each sentence		Automatic
Referring Expressions	expressions that refer		Manual
Referent Attributes	unchanging properties of referents	Yes	Manual
Co-reference Bundles	referring expressions that co-refer		Manual
Time Expressions	TimeML times		Manual
Events	TimeML happenings and states		Semi-Automatic
Temporal Relationships	TimeML time and event order		Manual
Context Relationships	static non-temporal relationships	Yes	Manual
Semantic Roles	PropBank verbal arguments		Semi-Automatic
Event Valence	impact of an event on the Hero	Yes	Semi-Automatic
Propp's *Dramatis Personae*	character classes	Yes	Manual
Propp's Functions	event categories	Yes	Manual

Table 4.1: Story Workbench representations used for this work. Five representations were developed from scratch. Three representations were annotated completely automatically, which meant that if an error was discovered in the output of the automatic analyzer, I hand-corrected it myself in all texts. Five representations were annotated semi-automatically, which means an analyzer produced an automatic first-pass guess at the annotations, which were then corrected by the annotators. The remainder of the annotations were done manually, meaning the annotations were created from scratch by the annotators.

the semantic representations.

Characters

While not strictly a representation, the characters of the text are the units against which all other representations in the Story Workbench are indexed. For the Russian folktales, translations for each tale to be included in the corpus were either located in extant English collections or commissioned from a bilingual speaker. For extant translations, the texts were first run through optical character recognition and then hand corrected to match the original text. Texts were wrapped to 78 characters per line with a blank line between paragraphs, and relevant metadata such as title, source, and copyright were inserted as a comment at the top of the text. Despite the care taken at this step, a few typographical errors remained. As these were discovered I corrected them myself across all versions of a text.

Tokens

The first representation to be calculated for each text was the token representation. Tokens are defined as simple, unbroken spans of characters. A token marks the location of each word or word constituent, following the Penn Treebank tokenization conventions (Marcus, Marcinkiewicz, and Santorini 1993). Importantly, this convention marks *n't* and *'ve* contractions as their own tokens, but leaves hyphenated words as one large token. This representation was automatically calculated by the Stanford tokenizer. Although the tokenizer is extremely accurate (greater than 99%), it still produced a few errors. As these were discovered I corrected them myself across all versions of a text. Example (4) shows a tokenization of a sentence, where each of the eight tokens is underlined.

(4) He would n't fight the three-headed dragon .

Multi-word Expressions

Multi-word expressions (MWEs) are words that are made up of multiple tokens. MWEs are important because many appear independently in sense inventories, and they must be marked (see §4.2.2) to attach word senses to them. Example (5) shows two types of continuous multi-words: a compound noun (*world record*) and a proper noun (*Guinness Book of World Records*).

(5) The world record is found in the Guinness Book of World Records.

MWEs may or may not have unrelated interstitial tokens. An example non-continuous MWE, the verb-particle multiword *look up*, is shown in (6).

(6) He looked$_1$ the word up$_1$ in the dictionary.

Although there are now detectors available for MWEs (Kulkarni and Finlayson 2011), there were none available when the corpus was being constructed. Thus the annotators were required to manually find and mark MWEs. This was not performed as a separate annotation task, but rather in the course of annotating word senses and semantic roles (see §4.2.2).

Part of Speech Tags

Each token and multi-word expression is tagged with a Penn Treebank part of speech tag (Marcus, Marcinkiewicz, and Santorini 1993). This representation was automatically calculated by the Stanford Part of Speech tagger (Toutanova et al. 2003). The tagger has an accuracy greater than 98%. The annotators corrected errors were corrected in the course of annotating word senses and semantic roles. Part of speech tags are one of the fundamental representations in the Workbench — they are important for identifying verbs for semantic role labeling, nouns for use in referring expressions, adjectives for attributes, and so forth.

Lemmas

Each token and multi-word expression that is not already in root form is tagged with its lemma, or root form. This is a simple annotation which merely attaches a string to the token or MWE. This representation was automatically calculated using a Java implementation of the Wordnet stemmer *morphy*. The stemmer is reasonably accurate, and errors were corrected by the annotators in the course of annotating word senses and semantic roles.

Sentences

Sentences are important when calculating parse trees, which themselves can be used to calculate higher level representations such as semantic roles. Sentences are merely lists of consecutive tokens, and they were automatically calculated by the Stanford sentence detector. The sentence detector is extremely accurate, but, if an error was discovered, I corrected these myself in all texts.

Context-Free Grammar Parse

The final syntax representation is a context-free grammar parse calculated by a statistical parser trained on the Penn Treebank. In this work the parses were generated by the Stanford parser (Klein and Manning 2003). Because the parses were primarily used to calculate semantic role labels, they were not corrected by annotators, even though they are included in the final corpus.

4.2.2 Senses

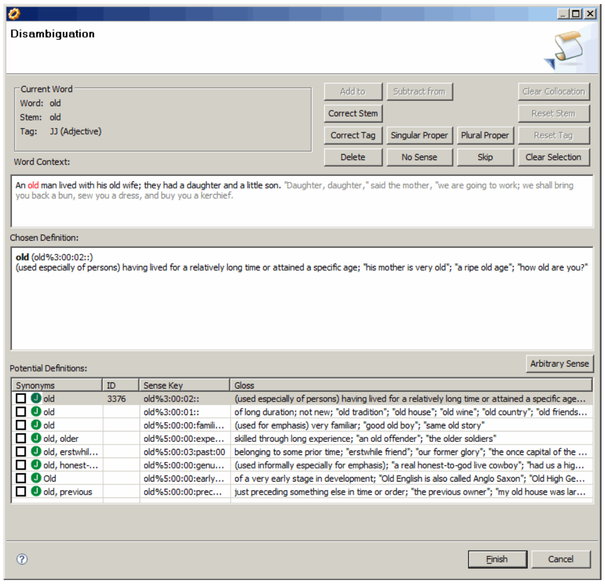
The second set of representations are those that encode the meanings of the words in the text. Meaning was captured in two representations: word senses and semantic roles.

Word Senses

Word sense disambiguation (WSD) (Agirre and Edmonds 2007) is a well-known NLP task. In it, each token or MWE is assigned a single sense from a sense inventory. For this work I used Wordnet 3.0. Because most WSD algorithms are not much better than the default most-frequent-sense baseline, this annotation was done completely manually by the annotators. While they were assigning word senses, they also corrected MWE boundaries, part of speech tags, and lemmas via a special dialog, shown in Figure 4-3. While Wordnet's coverage is excellent, it occasionally lacks an appropriate word sense. In those cases, the annotators found a reasonable synonym and substituted that sense. In the rare case that they could not find an appropriate substitute, annotators were allowed to mark "no appropriate sense available."

Semantic Roles

The second aspect of word meaning was verb argument structure. This is the well known semantic role labeling task as described in PropBank (Palmer, Kingsbury, and Gildea 2005). This annotation was done semi-automatically, the automatic analysis being done by a basic statistical semantic role labeler modeled on the analyzers described in (Pradhan et al. 2005) and (Gildea and Jurafsky 2002). This labeler was run over the texts to create argument boundaries and semantic role labels for each verb. Each verb was also assigned a PropBank *frame*, which is a list of allowed roles and their descriptions. The identity of this frame was the only piece of information not automatically annotated by the labeler. Annotators were required to add the frame id, missing arguments and semantic role labels, and to correct the extant argument boundaries and labels. As was the case for word senses, sometimes an appropriate frame was not available in the PropBank frame set. This happened more frequently (perhaps once or twice per text) than word senses on account of the much smaller size, and lesser coverage, of the PropBank frame set. In these cases the annotators found the closest matching frame and assigned that instead.



Current Word
Word: old
Stem: old
Tag: JJ (Adjective)

Word Context:

An old man lived with his old wife; they had a daughter and a little son. "Daughter, daughter," said the mother, "we are going to work; we shall bring you back a bun, sew you a dress, and buy you a kerchief.

Chosen Definition:

old (old%3:00:02::)
(used especially of persons) having lived for a relatively long time or attained a specific age; "his mother is very old"; "a ripe old age"; "how old are you?"

Potential Definitions:

Synonyms	ID	Sense Key	Gloss
□ old	3376	old%3:00:02::	(used especially of persons) having lived for a relatively long time or attained a specific age…
□ old		old%3:00:01::	of long duration; not new; "old tradition"; "old house"; "old wine"; "old country"; "old friends…
□ old		old%5:00:00:famili…	(used for emphasis) very familiar; "good old boy"; "same old story"
□ old, older		old%5:00:00:expe…	skilled through long experience; "an old offender"; "the older soldiers"
□ old, erstwhil…		old%5:00:03:past:00	belonging to some prior time; "erstwhile friend"; "our former glory"; "the once capital of the …
□ old, honest-…		old%5:00:00:genu…	(used informally especially for emphasis); "a real honest-to-god live cowboy"; "had us a hig…
□ Old		old%5:00:00:early…	of a very early stage in development; "Old English is also called Anglo Saxon"; "Old High Ge…
□ old, previous		old%5:00:00:prec…	just preceding something else in time or order; "the previous owner"; "my old house was lar…

Figure 4-3: The Story Workbench WSD dialog box that allows correction of part of speech tag, stem, MWE boundaries, proper noun markings, and word senses.

Label	Usual Meaning
ARG0	subject, agent, or theme
ARG1	object or patient
ARG2	instrument
ARG3	start state or starting point
ARG4	benefactive, end state or ending point
ARG5	direction or attribute
ARGM	modifying argument, usually augmented with a feature
ARGA	agentive argument where the agent is not ARG0; see Example (7)

Table 4.2: PropBank labels and their usual meanings. These meanings are abstracted across the different frames in PropBank's frame set.

The PropBank annotation scheme assigns a set of arguments (spans of tokens, potentially discontinuous) to each verb along with a primary category role for each argument, called the label, which is one of the tags listed in Table 4.2.2.

An agentive argument is used in the case when the ARG0 argument is not the agent of the action, as in verbs that take causative constructions, such as (7).

(7) [Mary]$_{\text{ARGA}}$ $\underline{\text{checked}}$ [John]$_{\text{ARG0}}$ into [a mental ward]$_{\text{ARG1}}$.

In addition to a label, each argument can be marked with a second tag, called the *feature*. Features mark an argument as fulfilling a common role for a verb, such as providing a direction, location, manner, negation, or modal, among others.

4.2.3 Entities

Because Propp's morphology is founded on event structure, namely, *who is doing what to whom, when*, we need to know the *who* and *whom*. The raw information for calculating this is given by the referring expression and co-reference representations. An overview of both representations is given here; for details, see (Hervás and Finlayson 2010).

Referring Expressions

The referring expression representation marks collections of tokens that refer to something, where the collection may be continuous or discontinuous. This representation was annotated manually. Two referential expressions are underlined in (8). In this sentence, both referents are people — concrete things in the story world.

(8) John kissed Mary.

While this simple example covers a large number of cases, they are far from the full story. Importantly, as in (9), referents may or may not have physical existence.

(9) John had an idea.

Here, the second referent is an abstract object. Similarly, as in (10), we can refer to things that don't exist.

(10) If John had a car$_1$, it$_1$ would be red.

The car does not exist, and yet we still refer to it.

Generally, if something is referred to using a noun phrase, it should be marked as a referent. This definition has the convenient property of having us to mark events (such as "kissed" above) only when they are picked out further beyond their use as a verb. Consider (11).

(11) John drove Mary to work.

In (11) there are three noun phrases, and normally one would not mark the driving event as a reference, in accordance with our intuition. But if there were a second sentence, as in (12):

(12) John drove$_1$ Mary to work. It$_1$ took forever.

The act of driving is being picked out, using the noun *it*, as something interesting to talk about above and beyond its mere mention in the story. This implies that the original event mention should be marked as a referring expression as well.

Co-reference Relationships

Example (10) also illustrates an important and obvious point, namely, that a single referent can be mentioned several times in a text. In (10) it is the car that is mentioned twice. In this case, there is a single referent (the car) with two referring expressions (the phrases *a car* and *it*). These last two referring expressions are *co-referential* because they refer to the same referent. As was done in (10), numeral subscripts indicate that the two referring expressions refer to the same referent.

To build referents out of referring expressions, collections of referring expressions that all refer to the same thing are brought together into a co-reference bundle. Therefore a co-reference bundle is just a list of referring expressions referring to the same thing with an attached name for ease of identification. This representation was annotated manually.

It is interesting to note that, while such a "bundling" representation is adequate for the purposes of this work, it is unsatisfactory from a cognitive point of view. There are a number of examples from our annotation work where it was unclear how a co-reference should be bundled. They include cases such as mistaken identity, future/unknown identity, imaginary referents, negation, transformation, and future states.

4.2.4 Timeline

To construct the timeline of the story, I used an established representation suite, TimeML (Pustejovsky et al. 2003). TimeML comprises three representations: time expressions, events, and time links. The first two mark the objects that populate the timeline, and the last defines the order of those objects on the timeline. Examples in the section are drawn from those in (Saurí et al. 2006).

Label	Description
Occurrence	happening that does not fall into another category
Reporting	someone declares something, narrates an event, informs about an event
Perception	physical perception of another event
Aspectual	specific aspect of event history, such the event's beginning or end
Intensional Action	occurrence from which we can infer something about an event argument
State	circumstance in which something obtains or holds true
Intensional State	refers to alternative or possible worlds

Table 4.3: TimeML event types and their meanings.

Time Expressions

Time expressions mark the location, type, and value of temporal expressions. Each expression is a sequence of tokens, potentially discontinuous, that indicate a time or date, how long something lasted, or how often something occurs. Temporal expressions may be calendar dates, times of day, or durations, such as periods of hours, days, or even centuries. Temporal expressions can be precise or ambiguous.

(13) The train arrived at 8:10 p.m. (Time)

(14) The train arrived on October 5th (Date)

(15) He was stuck on that island for almost a year. (Duration)

(16) Every day he gets a caffe americano double-soy no-sugar. (Set)

In addition to its constituent tokens, annotators also marked each expression with a tag that indicated if it was a *Date*, a *Time*, a *Duration*, or a *Set* of one of those three. Interestingly, time expressions are extremely sparse in folktales, with only 142 instances over the whole corpus of 18,862 words, averaging to only 7.5 time expressions per 1,000 words. Indeed, most of the tales had fewer than 10 time expressions, and two had only a single one. This unexpected fact is perhaps due to folktales generally occurring on unspecified dates, or altogether outside of history. Regardless of the reason, time expressions proved to have little importance for the timelines as a whole. This representation was annotated manually.

Events

Because events are central to Propp's morphology, I go over the event representation here in a fair amount of detail. Events are defined as happenings or states. They can be punctual, as in (17), or they can last for a period of time, as in (18). For the most part, circumstances in which something obtains or holds true, such as *shortage* in (19), are considered events.

(17) A fresh flow of lava, gas and debris erupted there Saturday.

(18) 11,024 people, including local Aeta aborigines, were evacuated to 18 disaster relief centers.

(19) Israel has been scrambling to buy more masks abroad, after a shortage of several hundred thousand.

Events are marked as one of seven different types, and these types are of critical importance for this work. They are listed in Table 4.3.

Occurrence This class is a catch-all for events that happen or occur in the world. Examples are:

(20) The Defense Ministry said 16 planes have landed so far with protective equipment against biological and chemical warfare.

(21) Mordechai said all the gas masks from abroad would arrive soon and be distributed to the public, adding that additional distribution centers would be set up next week.

Reporting These events are an occurrence where a person or an organization declares something, narrates an event, informs about an event, etc. Some examples of verbs that fall into this class are *say*, *report*, *tell*, *explain*, *state*, and *cite*.

(22) Punongbayan said that the 4,795-foot-high volcano was spewing gases up to 1,800 degrees.

(23) No injuries were reported over the weekend.

Perception These events are any occurrence that involves the physical perception of another event. Perception events are typically expressed by verbs like *see*, *watch*, *glimpse*, *behold*, *view*, *hear*, *listen*, or *overhear*.

(24) Witnesses tell Birmingham police they saw a man running.

(25) ”You can hear the thousands of small explosions down there”, a witness said.

Aspectual In languages such as English and French there is a grammatical device of aspectual predication, which focuses on different facets of event history, such as initiation, re-initiation, termination, culmination, and continuation. Thus an aspectual event is an occurrence that concerns the beginning, ending, or other period of time as part of an event.

(26) The volcano began showing signs of activity in April for the first time in 600 years.

(27) All non-essential personnel should begin evacuating the base.

Intensional Action An I-Action is a type of occurrence that introduces an event argument (which must be in the text explicitly) describing an action or situation from which we can infer something given its relation with the I-Action. For instance, the events introduced as arguments of the actions in (28) and (29) (surrounded by square brackets) have not occurred or will not necessarily occur when the I-Action takes place. Explicit performative predicates, as in (30) are also included here. Note that the I-Action class does not cover states – these are covered by Intensional States, see below.

(28) Companies such as Microsoft or a combined Worldcom MCI are trying [to monopolize] Internet access.

(29) Palestinian police [prevented] a planned pro-Iraq rally by the Palestinian Professionals' Union.

(30) Israel will ask the United States [to delay] a military strike against Iraq.

State These describe circumstances in which something obtains or holds true. Because everything is always in one state or another, annotators did not annotate all possible states, only those states that are identifiably changed over the course of the story. For instance, in (31), in the expression *the Aeroflot Airbus* the relationship indicating that the Airbus is run and operated by Aeroflot is not a state in the desired sense. Rather, because it is persistent throughout the event line of the document, it is not marked up as a TimeML expression, but rather as a Context Relationship (§4.2.5).

(31) All 75 people on board the Aeroflot Airbus died.

Intensional State An Intensional State (I-State) event is similar to the I-Action event. This class includes states that refer to alternative or possible worlds, (delimited by square brackets in the examples below), which can be introduced by subordinate clauses (32), untensed verb phrases (33), or nominalizations (34). All I-States were annotated, regardless of whether they persisted throughout the whole text.

(32) “We believe that [his words cannot distract the world from the facts of Iraqi aggression].”

(33) “They don't want [to play with us],” one U.S. crew chief said.

Temporal		Aspectual	Subordinating
Simultaneous		Initiates	Modal
Identity		Culminates	Factive
After	Before	Terminates	Counter-Factive
Immediately After	Immediately Before	Continues	Evidential
Includes	Included by	Re-initiates	Negative Evidential
During	Inverse During		Conditional
Begins	Begun By		
Ends	Ended By		

Table 4.4: TimeML Temporal Link categories and types.

(34) ”We're expecting [a major eruption],” he said in a telephone interview early today.

Additional Information In addition to the main class of the event, annotators marked the polarity and modality of each event. Thus if an event is negated, as in (35), it was marked as negative polarity.

(35) The US did not attack.

Time Links

A time link is a relationship between two times, two events, or an event and a time. It indicates that a particular temporal relationship holds between the two, for example, they happen at the same time, as in (36), or one happens for the duration of the other, as in (37). Other less intuitive examples of time links between two events include if one event is temporally related to a specific subpart of another event, as in (38), or imposes a truth-condition on another event, as in (39). This representation was annotated manually.

(36) The train arrived after 8:10 p.m. (Temporal)

(37) He was stuck on that island for almost a year. (Temporal)

(38) John started to read. (Aspectual)

(39) John forgot to buy some wine. (Subordinating)

Time links fall into three major categories, each of which has a number of subtypes, as listed in Table 4.4. **Temporal** links indicate a strict ordering between two times, two events, or a time and an event, as in (36) and (37). Six of the temporal links are inverses of other links (e.g., *After* is the inverse of *Before*, *Includes* is the inverse of *Included By*, and so forth). Annotators used one side of the pair preferentially (e.g., *Before* was preferred over *After*), unless the specific type was specifically lexicalized in the text, e.g., the annotators would mark an *After* link for (36). **Aspectual** links indicate a relationship between an event and one its sub-parts, as in (38). **Subordinating** links indicate relationships involving events that take arguments, as in (39). Good examples are events that impose some truth-condition on their arguments, or imply that their arguments are about future or possible worlds. The differences between these link types were extremely important to deriving the timeline of stories, as described in Section 5.1.

4.2.5 New Representations

I defined three new representations to capture important aspects of the semantics of stories that were not covered by any other established annotation scheme. They were referent attributes, context relationships, and event valencies.

Type	Description
Physical	visible or measurable characteristics such as size, height, and weight
Material	what a referent is made or composed of, or one of its ingredients
Location	identifying spatial position of a referent, e.g., "His front teeth"
Personality	non-physical character traits of characters
Name/Title	nicknames, proper names, titles, and other terms of address
Class	answers the question "What kind?"
Origin	whence an object comes, e.g., "Cockroach milk"
Whole	what the referent is (or was formerly) a part of
Ordinal	indicates the order or position of the referent in a set
Quantification	answers the question "Which one(s)?"
Mass Amount	answers the question "How much?"
Countable Amount	specific numbers that answer the question "How many?"
Descriptive	catch-all for attributes that do not fall into another category

Table 4.5: Categories of referent attributes and their meanings.

Referent Attributes

A referent is something that is being referred to in the text (see §4.2.3), and an attribute is anything that describes a referent. An attribute can be delivered in a copular construction, such as in (40), or as part of a compound noun phrase, as in (41). Importantly, attributes are defined to be *permanent* properties of the referent in question, meaning that they should not change over the timeline of the story. If they did change, they were considered TimeML states, and were annotated as such (see §4.2.4).

(40) John is brave.

(41) The red ball.

Each attribute was additionally assigned one of the tags listed in Table 4.5. This tag allowed the post-processing to augment the description of the referent appropriately. Because this representation was newly developed for this work and no automatic analyzers were immediately available, this representation was annotated manually.

Context Relationships

This representation marked static relationships between referents in the text. Like the semantic role representation, a particular expression was marked as anchoring the context relationship, such as *siblings* in (42). Referents participating in that relationship were marked with roles relative to the anchor. Role marking could be a Wordnet sense or a PropBank role, as the annotators saw fit. In (42) *Jack* would be marked with the Wordnet sense for *brother* and Jill with the sense for *sister*. Implicit relationships (i.e., without an anchor) could be marked as well, as in (43), where the fact that *They* is equivalent to the set {*Jack*,*Jill*} can be marked by tagging *They* with the Wordnet sense for *set*, and both *Jack* and *Jill* with the sense for *member*.

(42) [Jack] and [Jill] were siblings.

(43) [Jack] and [Jill] went up the hill. [They] fetched a pail of water.

Allowing annotators to mark roles with either Wordnet senses or PropBank roles allowed the representation to cover relationships not only where there was a specific instantiation of the role in the lexicon, but also relationship roles that were more easily expressed as a role to a verb. For example, in (44), *chicken legs*, which fills the role of *thing stood on* might be termed the *prop* or *helper*, but what about the hut? Is there a good noun meaning *thing being held up*? Even if we can find an appropriate single sense in Wordnet to cover this particular role, there is no guarantee that we will

Valence	Description	Example
-3	immediately bad for the hero or his allies	the princess is kidnapped; the hero is banished
-2	may lead directly to a -3 event	the hero and the dragon fight to the death
-1	someone threatens an event that would be -2 or -3	the witch threatens death to, or chases, the hero
0	neither good nor bad	
+1	someone promises an event that would be +2 or +3	an old man promises to help someday when the hero most needs it
+2	may lead directly to a +3 event	someone hides the hero from pursuit
+3	immediately good for the hero or his allies	the hero marries the princess; the hero is given gold from the treasury

Table 4.6: Event valencies and their meaning.

be able to find one for every role in a relationship.

(44) ... [a little hut] that stood on [chicken legs]...

Because this representation was developed specifically for this work, automatic analyzers were not available, and therefore this representation was annotated manually.

Event Valencies

Event valency is akin to Wendy Lehnert's positive or negative mental states (see §3.4.3). This representation indicates how positive or negative an event is for the hero. The scale ran from -3 to +3, including 0 (neutral) as a potential valence, rather than being restricted to just positive or negative as in Lehnert's representation. The import of each valence on the scale is laid out in Table 4.6). This representation was annotated manually.

This information was included to fill a specific gap in the representation suite. In the course of developing the Lehnert Plot-Unit example (§3.4.3, we noted the importance of inferred event valences to discovering the correct plot units. Many times the information that was directly incorporated into the final plot unit pattern was not explicitly mentioned in the story text itself, but was rather inferred by the Genesis commonsense reasoner. Constructing an equivalent module that would work with the wide-ranging semantics of the Russian folktales is a daunting research task. Commonsense reasoning is topic of active research, where even state-of-the-art inference engines and databases are not equal to the task of inferring the information needed, even in this simple context. Pursuing such a reasoner would have taken me far afield of the central concern of this work, namely, extraction of narrative structure. Instead I approached this as a annotation task, much like all the other information collected in the corpus.

4.2.6 Propp's Morphology

Finally, Propp's morphology was translated into two representations, one for the *dramatis personae*, and one for the functions.

Dramatis Personae

As described in Chapter 2, Propp identified seven types of character found in his folktales. I originally intended these annotations for use as a gold standard against which to measure character clusters extracted from the final morphology. However, layering another clustering problem on top of the clustering problem already in hand (function categories and transitions) proved to be too difficult at this stage of the work. Consequently the *dramatis personae* annotations were incorporated into

Character	Description
Hero	main character of the story
Villain	perpetrator of the villainy; struggles with the Hero
Helper	accompanies and assists the Hero
Donor	prepares and provides the magical agent to the Hero
Princess	sought-for person, not necessarily female
Dispatcher	sends the Hero on his adventure
False Hero[†]	someone who pretends to be the Hero to gain the promised reward

Table 4.7: Propp's *dramatis personae* and their meanings. [†]Did not appear in the corpus.

the data provided to ASM and used to help derive the morphology structure. I leave the derivation of the *dramatis personae* types for future work.

This representation consisted of seven labels, listed in Table 4.7. Any number of these could be attached to a particular referent in the text. Not all characters fulfilled a *dramatis personae* role, and in such cases no tag was attached to that referent. In other cases, as Propp noted, a single character fulfilled more than one role. This representation was annotated manually.

Functions

The final representation marked Propp's functions. This annotation served as the standard against which the results of the Analogical Story Merging algorithm were measured.

Annotating Propp's functions was a delicate task. While Propp described his morphology in great detail, it still was not specified in such a way as to allow unambiguous annotation in text. Propp's monograph is an enlightening read, but it is not an effective annotation guide. There are at least four main problems with Propp's scheme as described: unclear placement, implicit functions, inconsistent marking of trebling, and, in a small number of cases, apparent disagreement between Propp's own categorization scheme and what is found in the tale.

With regards to unclear placement, consider, for example, the following excerpt of Afanas'ev's tale #148.

> The tsar went in person to beg Nikita the Tanner to free his land from the wicked dragon and rescue the princess. At that moment Nikita was currying hides and held twelve hides in his hands; when he saw that the tsar in person had come to see him, he began to tremble with fear, his hands shook, and he tore the twelve hides. But no matter how much the tsar and tsarina entreated him, he refused to go forth against the dragon. So they gathered together five thousand little children and sent them to implore him, hoping that their tears would move him to pity. The little children came to Nikita and begged him with tears to go fight the dragon. Nikita himself began to shed tears when he saw theirs. He took twelve thousand pounds of hemp, tarred it with pitch, and wound it around himself so that the dragon could not devour him, then went forth to give him battle.

Propp indicates the presence of functions B and C. Propp defines B as "Misfortune or lack is made known; the hero is approached with a request or command; he is allowed to go or he is dispatched." with an abbreviated definition of "mediation, the connective incident". He defines C as "The Seeker agrees to or decides upon counteraction," with an abbreviated definition of "beginning counteraction." Roughly, these two functions are the presentation of the task to the hero (B), and the acceptance of that task (C).

Where exactly is B? Is it the whole section? Is it from the word *entreated* to the word *begged*? Should function boundaries correspond to sentence or paragraph boundaries? Are the children *dramatis personae* in this tale (*Dispatchers*?), or are they are merely instruments of the tsar and tsarina? Is their *imploring* to be considered part of B?

Annotators marked two groups of tokens when identifying functions. First, they marked a region which captured the majority of the sense and extent of a function. This was usually a sentence, but extended to a paragraph or more in some cases. Second, they marked a defining word for the function, which usually took the form of single verb. In cases where that single verb is repeated close by the first marking, and refers to the same action, these repeats were marked as well. In the case above, annotators marked the region "the tsar and tsarina entreated...begged him with tears to go fight the dragon." as B, and picked the verb "entreated" as the defining verb.

Where exactly is C? This is the decision to go forth against the dragon. It seems to happen somewhere between Nikita's shedding of tears and his preparation for battle by obtaining hemp, but it is not expressed anywhere directly in words; that is, the function is *implicit*. Propp notes that implicit functions occur frequently, yet he gives no way to identify when they happen, and marks them inconsistently. When the annotators could find no explicit mention of a particular function that Propp, in his Appendix III, indicated occurred in a tale, they chose the most closely logically related event and marked it either as an *Antecedent* or a *Subsequent*, as appropriate. For C in the section above, the region was the sentence "Nikita himself began to shed tears when he saw theirs." and "shed" was marked as the defining verb. This function was marked as an *Antecedent*.

With regards to inconsistently marked trebling (function groups that were repeated two, three or four times in succession) or when indicated functions did not seem to match the tale itself, the annotators did their best to determine the correct marking. Fortunately, most of the time, typographical errors were restricted to disagreement in function subtypes which does not directly impact this results presented in this work.

4.3 Annotation Evaluation

The quality of the annotations can be assesed by measuring the inter-annotator agreement. I describe briefly the annotation procedure, and review the annotation agreement measures for each representation.

In cases where an established representation was being annotated, I prepared an annotation guide from the available material for the annotation team. An annotation team comprised two annotators and an adjudicator. The adjudicator was either an annotator already experienced in that representation, or myself (if no other adjudicator was available). After the annotation of the same few thousand words (2-3 texts) by the two annotators, the whole annotation team met and used the merge tool described in Section 4.4.3 to bring the annotations together into a single document, which was then corrected by discussion guided by the adjudicator. This process was repeated until all the texts were annotated.

In cases of representations developed anew for this work, I adjudicated the annotations, and created the annotation guide from scratch as the team progressed.

The most uniform measure of agreement across the different representations is the F_1-measure, which is calculated in the standard way (van Rijsbergen 1979). I used the F_1-measure instead of the more common Kappa statistic (Carletta 1996) because of the difficulty of calculating the chance-level of agreement for most of the representations. The F_1-measure is a natural outgrowth of the merge process, has a clear interpretation with regard to the data, and allows a more direct comparison between different representations. Table 4.8 summarizes the agreements for the different representations annotated either manually or semi-automatically.

Overall the agreement values are good. They start extremely high for the low-level syntactic representations, where the automatic analyzers have high accuracy. The agreement naturally declines for more complex and semantically meaningful representations.

Three representations required different agreement measures to properly reflect their quality. First, the co-reference agreement is not properly measured with a strict F_1-measure. This is illustrated as follows. Consider the five referring expressions are underlined in Example (45), and two different markings of co-reference bundles in Table 4.9.

(45) $\underline{\text{John}}$ and the $\underline{\text{dog}_1}$ played ball. $\underline{\text{He}_1}$ would throw, and the $\underline{\text{dog}_2}$ would catch. $\underline{\text{He}_2}$ was good.

Representation	New?	Measure	Agreement
Tokens		n/a	-
Multi-word Expressions		F_1-measure	0.68
Sentences		n/a	-
Part of Speech Tags		F_1-measure	0.98
Lemmas		F_1-measure	0.93
Word Senses		F_1-measure	0.78
CFG Parse		n/a	-
Referring Expressions		F_1-measure	0.91
Referent Attributes	Yes	F_1-measure	0.72
Co-reference Bundles		chance-adjusted Rand index	0.85
Time Expressions		F_1-measure	0.66
Events		F_1-measure	0.69
Temporal Relationships		F_1-measure	0.59
Context Relationships	Yes	F_1-measure	0.54
Semantic Roles		F_1-measure: ARG[0-5] only	0.60
Event Valence	Yes	F_1-measure	0.78
Proppian *Dramatis Personae*	Yes	F_1-measure	0.70
Proppian Functions	Yes	F_1-measure: region overlap	0.71

Table 4.8: Agreement measures for each representation.

Referent	Annotator 1	Annotator 2
John	*John*-*He*$_1$-*He*$_2$	*John*-*He*$_1$
Dog	*dog*$_1$-*dog*$_2$	*dog*$_1$-*dog*$_2$-*He*$_2$

Table 4.9: Example of co-reference bundles marked by two annotators.

One annotator marks *John*-*He*-*He* and *dog*-*dog*, but the other marks *John*-*He* and *dog*-*dog*-*He*. A strict F_1-measure, operating on the co-reference bundles themselves, would score this as zero agreement, because no co-reference chain in one set of annotations exactly matches any chain in the other set of annotations. Nevertheless, there is some agreement: both annotators correctly marked two co-reference relationships, that between *John* and the first *He*, and that between the two instances of the word *dog*. A better measure that reflects this partial agreement is the chance-adjusted Rand index (Hubert and Arabie 1985) which, for this particular example, nets a score of 1/6 rather than zero.

Second, a less strict F_1-measure was used to assess the semantic role annotations. The exact measure is an extremely high bar for such a complex representation: it nets only 0.36 averaged across all texts. This is because there is so much to annotate in this representation: an average verb usually has at least two arguments, often up to five. Each argument has two tags (its primary label and an auxiliary feature), and usually several words. Each verb also has five syntactic features. If an annotator disagrees with his counterpart on any one of these dimensions, this counts as a miss. A more representative agreement measure is to ignore agreement between argument auxiliary features, the verb syntactic features, and arguments other than the core arguments (these were not used in the final analysis) which produces an agreement of 0.60.

Third, Propp's functions needed a special agreement measure that took into account the difficulty of translating Propp's monograph into a consistent annotation guide. Propp's monograph was not intended as an annotation at all, but rather a theory of narrative put forth in the 1920's before there was even such a thing as computational linguistics (or, indeed, computers). Propp's theory is difficult to translate into a precise annotation specification. He was quite vague about the identity of many of his functions; his Appendix III has numerous inconsistencies and vagaries. These problems result in a strict F_1-measure agreement of only 0.22, quite low. This number did not seem to reflect the annotation team's intuition that agreement was actually fair to good, once minor variations were ignored. Instead of a strict measure, then, I formulated a more generous measure in which two function markings were considered to agree if there is a substantial (more than half) overlap in the function regions. This nets an agreement to 0.71, more in line with the team's observation that the annotators did actually agree in broad outlines.

In four cases the agreement falls below 0.7. First, time expressions achieved a 0.66 F_1-measure. This is not especially worrisome because time expressions are quite sparse and were generally superfluous to the actual progression of the timeline. Second, context relationships achieved only an F_1-measure of 0.54. This is a bit troublesome, but this representation was primarily used to substitute individuals for the groups of which they were a part; other types of relationships were not used in the final analysis. Moreover, the final timelines were inspected manually to make sure all participants in the final event representations were individuals, and when an agglomerate object was discovered, the annotations were corrected as appropriate. So it is unlikely that this lesser agreement had a substantial effect on accuracy of the results. Both time links and semantic roles achieved F_1-measures of about 0.6. While these numbers are disappointingly low, these were also the two most complex representations in the suite. These lower agreement numbers naturally reflect the fact that the annotators had difficulty keeping all the complexities of these representations in mind. To improve the agreement in future annotation projects, annotator training and supervision needs to be improved, the Story Workbench facilities for visualizing and checking these representations should be expanded, and the representations themselves should be broken down into more manageable units.

4.4 The Story Workbench: Design Details

Here I provide more details on the design and implementation of the Story Workbench tool.

4.4.1 Design Principles & Desiderata

What principles and desiderata guided the design and implementation of the Story Workbench? When possible, I followed three principles:

1. Build on top of popular tools with large user and developer communities
2. Use open-source and freely-available programming libraries
3. Adopt widely-used and well-documented standards

In application design things change quickly. Today's hot new tool or language is tomorrow's ancient history. All three principles help insulate the work from the inexorable advances in the state of the art. The more one adheres to these principles, the more likely one's tools or languages are to evolve to the next stage, rather than be thrown out and replaced *in toto*. Similarly, standards usually have precise specifications and the backing of some independent organization, and are unlikely to suddenly disappear forever. If the tools and libraries are freely modifiable, then users need not be constrained by the features provided in the initial implementation, but can implement their own. This is especially important if the tool becomes popular. Finally, a benefit of all three principles is that the more popular a tool, library, or standard, the more resources (such as reference books or tutorials) are available to help users and developers.

In addition to the three principles, there were at least five desiderata:

1. The storage format should be human-readable and tool-independent.
2. The storage format should be modular and extensible.
3. The tool framework should be highly functional, modular, and extensible.
4. The tool should not commit to a specific operating system.
5. The tool should not commit to specific NLP technologies.

Data Format

The storage format should be human-readable and tool-independent Perhaps the most important design decision was the format of the data produced by the Story Workbench. This data, if useful at all, will long outlive any implementation. Thus it was extremely important that the data format be based on a well-documented and easily-understood standard. It is also imperative that the specification of the format be independent of any implementation. Years later a user needs to be able to understand the data without the benefit of software, and, in principle, recreate an interface to it from scratch.

The data should also be human-readable and, thus, in principle, editable using only a text editor. In the course of research small tweaks need to be made and data needs to be reviewed at a glance. Sometimes a working tool is not available to look at the data. This rules out binary formats and restricts us to a standard character set.

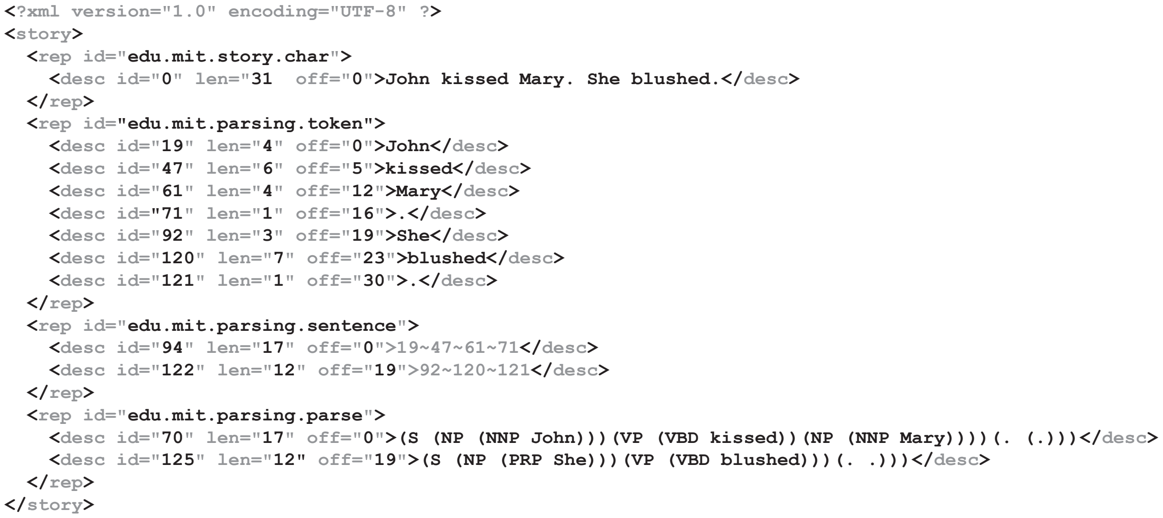
To satisfy this desideratum, I used the eXtensible Markup Language (XML) (Bray et al. 2006) as the base format for the files. XML satisfies both of the above properties: it is a widely-supported, easily-understood international standard, and it is human readable and editable by text editor alone. XML includes mechanisms for formally describing the well-formedness of XML files in its definition. One can check to see if a particular XML file is well-formatted with respect to a particular expectation.

The storage format should be modular and extensible The second important constraint on the data format was that it needed to admit additions of as-yet-unforeseen pieces; in other words, it needed to be *extensible*. I discussed above how there are many different aspects of meaning to a particular text. It is not possible to code them all at once, especially at our current level of knowledge. We should be able to annotate what is needed right now, and then, if later another aspect of meaning is needed, the format should accommodate the layering on top of what has already been done. This holds true, also, for end-users: if they want to take the tool and annotate a new aspect of meaning, they should be able to do this without redesigning the whole storage format.

Figure 4-4 is a concrete example of an actual data file, with some of the less important aspects trimmed for ease of exposition. The topmost tag has a number of representation blocks as children. The first representation block is the text itself, and is the only required representation. The following representation blocks are the different aspects of meaning that have been annotated for this text. The second representation block is the list of tokens for the text. The third representation block is the sentence representation, indicating the extent of each sentence. The final block gives a syntactic analysis of each sentence in a Penn Treebank style format. Each representation block has zero or more description children that each contain an atomic piece of data. Each description has a number of different XML attributes, the three most important of which are illustrated in the figure. The first is a unique identification number that can be used by other descriptions elsewhere in the document to refer to that description. The second and third are the offset and length of the description, which indicate the span of character indices the description covers. Between the opening and closing description tags is a string of characters or no characters at all that represent the data for that description. The format of this string is specific to each representation. This format is modular in that blocks (either whole representations, or individual descriptions) can be added or deleted without modifying other blocks.

Application Framework

The tool framework should be highly functional, modular, and extensible Along with modularity of the data format should come modularity of the Story Workbench itself. Because the XML data format outlined above allows anyone to define his own representations, the application framework on which the Story Workbench is based should support the creation of the infrastructure for producing guesses and checking constraints for that representation, as well as any special graphical user interface widgets that are required. These parts should be wrapped in a module that can be distributed by itself and can be added to any other instance of the Story Workbench with minimal trouble.



```
<?xml version="1.0" encoding="UTF-8" ?>
<story>
  <rep id="edu.mit.story.char">
    <desc id="0" len="31  off="0">John kissed Mary. She blushed.</desc>
  </rep>
  <rep id="edu.mit.parsing.token">
    <desc id="19" len="4" off="0">John</desc>
    <desc id="47" len="6" off="5">kissed</desc>
    <desc id="61" len="4" off="12">Mary</desc>
    <desc id="71" len="1" off="16">.</desc>
    <desc id="92" len="3" off="19">She</desc>
    <desc id="120" len="7" off="23">blushed</desc>
    <desc id="121" len="1" off="30">.</desc>
  </rep>
  <rep id="edu.mit.parsing.sentence">
    <desc id="94" len="17" off="0">19~47~61~71</desc>
    <desc id="122" len="12" off="19">92~120~121</desc>
  </rep>
  <rep id="edu.mit.parsing.parse">
    <desc id="70" len="17" off="0">(S (NP (NNP John)))(VP (VBD kissed))(NP (NNP Mary))))(. (.)))</desc>
    <desc id="125" len="12" off="19">(S (NP (PRP She)))(VP (VBD blushed)))(. .)))</desc>
  </rep>
</story>
```

Figure 4-4: Example story data file in XML format.

The tool should not commit to a specific operating system The Story Workbench should also run on many different platforms. There are many popular operating systems in use today and if the tool is to be widely adopted, it has to run on as many of those as possible.

I have satisfied these two desiderata by choosing the Java programming language (Gosling et al. 2005) and the Eclipse Application Development Framework (Gamma and Beck 2004) as the fundamental building blocks of the Story Workbench. Java is an inherently cross-platform language and runs on almost every operating system. It is open-source, freely-available, and in extremely wide use. There are numerous libraries and utilities available for Java, not just for natural language processing, but for other tasks as well. The Eclipse platform satisfies our demands for both functionality and modularity. Eclipse is open-source, freely-available, implemented almost entirely in Java, and has a large base of users and developers who are constantly at work to improve the tool. It brings ready implementations of the many aspects vitally important to a modern graphical application. It is built out of a set of *plugins*, modules that form a coherent piece of code that add a specific functionality on top of the base program. In the context of the Story Workbench, if end-users want to add a new representation or a suite of related representations, they would program their code against the Eclipse API and then package it as a plugin, which can then be distributed to and installed by any user of the Story Workbench. Indeed, the Story Workbench itself is only a collection of plugins that can be distributed to and run by any user of the Eclipse platform.

NLP Technologies

The tool should not commit to specific NLP technologies The Story Workbench makes a commitment to the semi-automatic annotation method. It does not commit to any particular type or implementation of the NLP technology it uses to perform guesses for the annotations. In this regard, the tool is technology-agnostic. To be sure, the default distribution provides implementations of specific tools. But the architecture of the tool is such that end-users can easily introduce new or different tools, just as they can introduce new representations, and when constructing a text can choose between all the different methods on hand. For example, at the time of this writing, the Story Workbench uses the Stanford Natural Language Group's Java tools to do the heavy lifting for low-level syntactic analysis. Nonetheless, there are many other tools that do the same sort of syntactic analysis that could conceivably take the place of the Stanford tools; these are easily integrated into the Workbench's plugin structure.

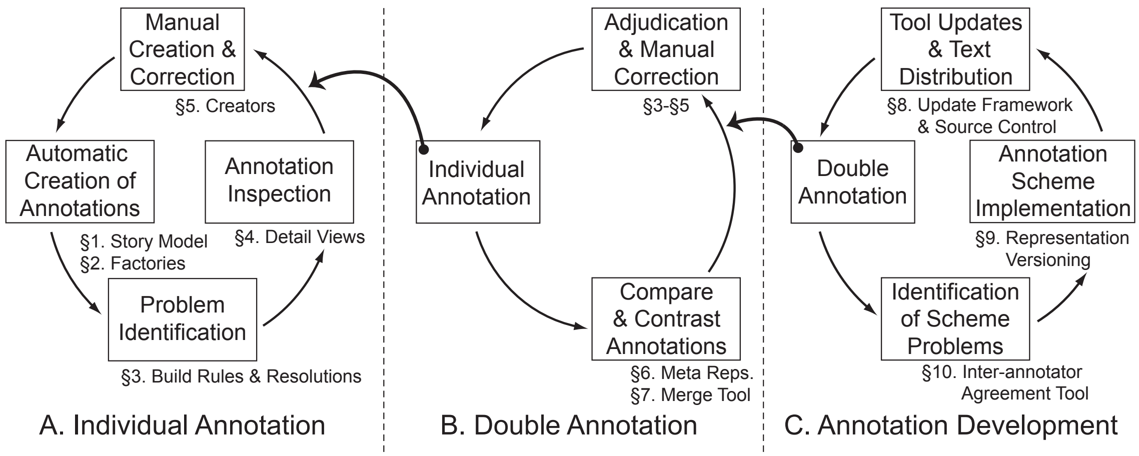

Manual Creation & Correction
§5. Creators
Automatic Creation of Annotations
Annotation Inspection
§1. Story Model
§2. Factories
§4. Detail Views
Problem Identification
§3. Build Rules & Resolutions
A. Individual Annotation
Adjudication & Manual Correction
§3-§5
Individual Annotation
Compare & Contrast Annotations
§6. Meta Reps.
§7. Merge Tool
B. Double Annotation
Tool Updates & Text Distribution
§8. Update Framework & Source Control
Double Annotation
Annotation Scheme Implementation
§9. Representation Versioning
Identification of Scheme Problems
§10. Inter-annotator Agreement Tool
C. Annotation Development

Figure 4-5: Three Loops of the Annotation Process. Letters correspond to subsections of §4.4.2. Number correspond to subsections of §4.4.3. Underneath each stage are the Story Workbench features that support that stage.

4.4.2 Annotation Process

Conceptually, the process of producing a gold-standard annotated corpus can be split into at least three nested loops. In the widest, top-most loop the researchers design and vet the annotation scheme and annotation tool. Embedded therein is the middle loop, where annotation teams produce gold-annotated texts. Embedded within that is the loop of the individual annotator working on individual texts. These nested loops are illustrated in Figure 4-5, and their letters refer to the section below.

A. Individual Annotation

Individual annotation occurs when an annotator annotates a text in a particular annotation scheme. This is the most fundamental component of the annotation process: it is in this tight loop that most of the effort in annotation is expended, and therefore is where most of the Story Workbench support is focused. Stages of this loop include automatic production of initial annotations, finding of problems with those annotations by either the annotator or the tool, and then manual creation, correction or elaboration of the annotations by the annotator. This loop is repeated for each text.

B. Double Annotation

Double annotation[1] is the norm when annotating complex representations or large amounts of text. It is done to guard against errors by allowing one annotator's results to be checked against another's. Although differences between annotations may be caused by simple mistakes, it is often the case that, for complex representations, the difference is legitimate and must be adjudicated by a third party. This adjudicator is responsible for comparing and contrasting the two annotation sets and correcting the discrepancies. Operationally, this loop involves distributing the texts to the individual annotators, comparing the resulting annotations, and discussing and correcting the differences. This loop repeats itself, either in parallel or in serial, for each text being annotated.

[1] I call it *double* annotation for ease of exposition; of course, in the general case, one may have as many annotators as one likes.

C. Annotation Development

The top-most loop is only necessary when one is developing a new representation, tool, or evaluating which texts to annotate. This loop involves developing the annotation scheme and annotation software, distributing the texts to the annotation teams (or individual annotators, if there is no double annotation), and finally evaluating the results to determine what changes should be made to the scheme or to the software.

4.4.3 Story Workbench Functionality

As indicated in the figure, the Story Workbench supports the three annotation loops with specific features, functions, and tools. Each section here is numbered to correspond with the figure.

F1. Story Model

The Story Workbench data structure for annotation distinguishes between *representations* and *descriptions*. A representation is the programmatic format of the annotation, while descriptions are actual annotations in that format. Within the Story Workbench, a representation is defined first by a data structure that specifies all the information possibly contained in a single annotation in that representation and second by a singleton lifecycle object that controls serialization and deserialization of descriptions for that representation, and defines on which other representations that representation depends. A description is a concrete instance of data that conforms to the representation specification and is linked to the representation lifecycle object.

The core of the Story Workbench is the story model, which is the data structure that describes a whole text and all of its annotations. Most importantly, it comprises maps from representation lifecycle objects to (1) sets of descriptions in that representation; (2) configuration information for that representation; and (3) an annotation factory for that representation (see §F2). The model has a primary, root representation, called the *character* representation, whose single description contains the actual characters for the text. All descriptions in the model are indexed to character offsets in the this singleton character description.

The two most important features of representations are that they are (1) layered, in that more complex representations can build upon simpler representations, and (2) extensible, in that new representations may be added. There are currently 18 representations implemented in the Story Workbench. These are discussed in detail in Section 4.2.

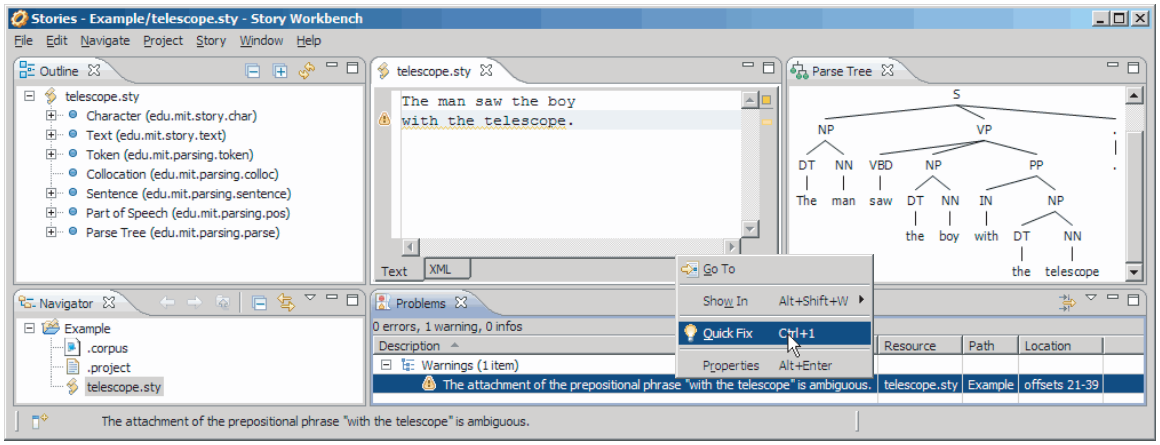
F2. Factories

One of the most important functions of the Workbench is the ability to provide automatic annotation. Automatic annotation is triggered by any change to the text or existing annotations, and are created by *annotation factories*, each of which is specific to a representation. An example of an annotation factory is a part-of-speech tagger (associated with the part of speech representation) which tags new tokens as they are created; a second example is a semantic role tagger that tags new verbs with semantic roles.

Factories are also responsible for keeping the model consistent: for example, if a character is inserted at the beginning of the text, all the annotations in the model, which are indexed to character offsets, must be shifted forward by one. This is taken care of automatically by the factories. Factories are extensible in that it is easy to add new factories to existing representations.

F3. Build Rules & Resolutions

Each representation has a number of *build rules* associated with it. Build rules are provided by the representation implementer; they check for known or suspected annotation errors. Build rules are run when the text is saved and the problems they find are displayed in the editor (highlighted for easy identification) and in separate views. In many of the annotation tasks for which the Story Workbench has so far been used, the task is set up so that the text starts with one problem per


Stories - Example/telescope.sty - Story Workbench
File Edit Navigate Project Story Window Help
Outline
telescope.sty
Character (edu.mit.story.char)
Text (edu.mit.story.text)
Token (edu.mit.parsing.token)
Collocation (edu.mit.parsing.colloc)
Sentence (edu.mit.parsing.sentence)
Part of Speech (edu.mit.parsing.pos)
Parse Tree (edu.mit.parsing.parse)
telescope.sty
Text
XML
Parse Tree
S
NP
VP
DT
NN
VBD
NP
PP
The
man
saw
DT
NN
IN
NP
the
boy
with
DT
NN
the
telescope
.
Go To
Show In Alt+Shift+W
Quick Fix Ctrl+1
Properties Alt+Enter
Navigator
Example
.corpus
.project
telescope.sty
Problems

```
The man saw the boy
with the telescope.
```

0 errors, 1 warning, 0 infos

Description	Resource	Path	Location
Warnings (1 item)			
The attachment of the prepositional phrase "with the telescope" is ambiguous.	telescope.sty	Example	offsets 21-39

The attachment of the prepositional phrase "with the telescope" is ambiguous.

Figure 4-6: Story Workbench problem and quick fix dialog.

potential annotation. The annotator's job is then to eliminate all the errors, which, when done, indicates the annotation in that representation is finished.

Problems found by build rules can be associated with problem resolutions. These resolutions are mini programs that, when run, fix the problem with which they are associated in a specific way. These resolutions are shown to the annotator when any graphical representation of the problem is clicked. As an example, for the MWE representation, it is considered an error to have two multiwords that share a token. Such an occurrence is marked by a red X in the problems list and the offending token or tokens are underlined in red in the editor. When the annotator clicks on either the underline or the X, a dropdown list appears with four resolution options: delete the first multiword, delete the second, remove the token from the first multi-word, or remove the token from the second. The annotator may choose any one of those resolutions or go about fixing the problem in his own way. Like most everything else, build rules and resolutions are extensible.

F4. Detail Views

Also associated with each representation is a detail view, which provides detailed information about the annotations for that representation. Detail views can provide information such as the internal structure of annotations or a detailed breakdown of field values with explanations. As an example, the detail view for the Semantic Role representation shows a list of all the verbs in the text; each verbs's semantic arguments are listed underneath, and clicking on the verb in the details view, or an argument to the verb, highlights the associated text in the editor. The detail viewer infrastructure is extensible: additional detail viewers can be added by third parties for existing representations.

One can also include in this category a number of specialized viewers and information transmission mechanisms that make the Story Workbench a user friendly environment. The *text hover* infrastructure allows representation implementers to show information in a tooltip when the mouse hovers over a specific section of the text. For example, when the mouse hovers over a word, a tooltip displays the part of speech tag and the root form (if the word is inflected). The Workbench also allows specialized viewers outside of the detail view — a good example of this is the parse tree viewer, which, when the cursor is placed on a sentence with a CFG parse, displays the parse in graphical tree form.

F5. Creators

The heavy lifting of annotation happens in the creator view, where the annotator can create and modify individual annotations. Each creator view is customized to its associated representation. For example, in the TimeML event representation, the creator view allows (1) identification of

Meta Rep	Description
Note	arbitrary text on an annotation to record observations, thoughts, ideas (§F7,9)
Origin	added upon description creation to record the identity of the creator (§F5)
Timing	added upon description creation to record creation and editing times (§F5)
Check	marks a problem as resolved or checked relative to a particular description (§F3)

Table 4.10: Important meta representations.

tokens involved in the event, (2) identification of head tokens, (3) marking the part of speech, tense, and aspect of verbal events, (3) identification of polarity tokens and associated polarity flag, (4) identification of tokens indicating cardinality and the number, and (5) identification of tokens indicating modality.

F6. Meta Representations

In addition to the main representations that encode the syntactic and semantic information of the text, there are so-called *meta* representations that track information common to all descriptions. Meta representations are also extensible. I identify four of the most important in Table 4.10. Perhaps the most useful of these is the Note meta representation, which allows annotators and adjudicators to record their thoughts and observations for later consideration.

F7. Merge Tool

A key function for double annotation and adjudication is the ability to compare and contrast annotations done by two different annotators. The Story Workbench provides a general tool for merging two sets of annotations into a single text. Each representation implementor provides a small amount of code that allows the Workbench to decide if two annotators are equivalent — this is provided in the implementation of the representation data structure (§F1). The tool then allows an adjudicator to decide which representations to merge, which should already be identical, and which should be ignored, and enforces constraints imposed by the representation hierarchy. The resulting annotated text can then be edited and corrected in the normal way (§F3-F5). The representation build rules may be defined so that, where possible, conflicting annotations from different annotators show up as errors to be corrected.

F8. Update Framework & Source Control

Integrated with the Workbench is a general means for interfacing with source control systems. Such systems, such as Subversion or CVS, can be used to great effect in annotation studies. Let us take Subversion as a concrete example. Researchers upload texts to be annotated to the Subversion repository. Annotators then *check out* the texts and do their annotation. The Workbench tracks when they have made changes, and displays the presence of those changes to the annotator; he then knows to *check in* his changes to the repository, where they are then available to the adjudicator or the researchers.

Also included in the Workbench is a sophisticated update functionality. This can be used to fix bugs or add new features without requiring a re-install of the tool.

F9. Representation Versioning

Developing an annotation scheme is facilitated by a versioning system for representation implementations. By way of example, imagine a researcher designs a representation that allows a single tag to be applied to any token. He implements the scheme, gives it a version number (say, 1.0), and has his annotators annotate some text with that representation. After considering the results (e.g., inter-annotator agreement scores, see §F10), he decides there should be a second tag to supplement the first. He then modifies the implementing code, increments the version (to 2.0), and specifies in the versioning system that when a v1.0 annotation is transformed into a v2.0 annotation, the new fields should be filled in with a tag called UNKNOWN. He writes a build rule that marks all annotations with an UNKNOWN tag as an error. He uses the update feature to distribute the new code to the annotators. When the annotators re-open their old texts, the Story Workbench automatically transforms the old annotations into the new, and they are presented with new errors that, when corrected, will naturally lead them to fill in the second field.

F10. Inter-annotator Agreement Tool

Finally, a general facility is provided for measuring inter-annotator agreements. The facility allows for automatic measurement of F_1-measures for all representations. There is also a facility that allows representation implementers to provide code allowing annotator agreements to be measured in terms of the Kappa score.

Chapter 5

Results

This chapter describes the primary contribution of the work: the application of the Analogical Story Merging (ASM) algorithm to learning the middle level of Propp's morphology. I first show how I transformed the corpus annotations into the data on which the algorithm was run. Then, I describe ProppASM, the particular implementation of ASM that was applied to that data to learn the morphology. Next, I detail the how the annotations of Propp's functions provided the learning target and analyze the results of the ProppASM algorithm against that target both quantitatively and qualitatively. Finally, I identify a number of shortcomings in the results and algorithm implementation and discuss how they may be addressed in future work.

5.1 Data Preparation

Section 3.3 described three key dimensions of the ASM algorithm: how to generate the initial model from the data, the allowed merge operations, and the prior distribution. In this section I describe how I generated the initial model from the corpus data.

5.1.1 Timeline

Because Propp's morphology is centered around the event structure, I constructed the initial model for the ProppASM implementation from each tale's timeline that was generated from the Story Workbench annotations. Although in general a timeline describes *who did what to whom, when*, a specific timeline can take many forms. Even in folktales, where the temporal order is relatively straightforward, events do not occur in a metronome-like lock-step. Events may be expressed out of order; they may be instantaneous or have duration; seconds, days, or years may pass between one event and the next; there are events that never happen, might happen, or will happen; and events occur simultaneously.

The most complete representation of such timeline information is not a line at all, but rather a graph indexed to some universal time axis. Such an approach might well be necessary for complex sorts of narrative, but the folktales in my corpus are quite simple in their temporal structure, and can be described by a linear timeline. All the events that actually occur in the story world can be unambiguously, and correctly, laid out in a definite order. Indeed, among the 15 tales in my corpus, only a single one contains a *meanwhile* construction (tale #127). This was of no consequence to tale's action, and I manually linearized that timeline without a problem.

For each tale, I constructed a linear timeline from the annotated TimeML events. Each timeline captures the order of the starting points of the events. To construct this sequence I wrote a simple algorithm that takes into account the different sorts of TimeML connections and arranges the events in the correct order. The algorithm proceeds as follows. First, it constructs a set of all unique events by examining *Identity* time links, which indicate that two event expressions co-refer. Second, the event sets are arranged into a graph where each vertex is an event set, and each directed edge represents a TimeML time link annotation. Each edge points forward in time from the starting point of one linked event to the starting point of the other. If the graph contained distinct subgraphs where one event was not reachable from another event by following links either backward or forward, then this was considered an error and the annotations were corrected. Third, the algorithm walks forward and backward in the graph, beginning with the first event mentioned and inserting events into the timeline before or after other events as appropriate. For example, if the algorithm is at event A and follows a *Before* link to another event B, then B must be placed on the timeline after A.

The procedure is complicated by two problems: one, there may be multiple events linked to a single event either forward or backward and the link types range in how close they "bind" two events together; and two, some events do not have any incoming links, but only outgoing links. I solved the first problem by defining an order of immediacy between the link types, shown in Table 5.1. The higher the immediacy, the closer the starting points of the two events on the timeline. For example, if there are two links *BEFORE(A,C)* and *IMMEDIATELY-BEFORE(B,C)*, then, because *Immediately Before* binds closer than *Before*, the resulting event order is A, B, C. I solved the second problem by allowing the algorithm to recurse backward along links.

Link Type	Immediacy
Before After	1
All Aspectual Links Immediately After Immediately Before	2
Ended By Ends Begins Begun By	3
Includes Is Included By During During Inverse	4
Simultaneous	5

Table 5.1: Table of immediacy rankings for TimeML temporal and aspectual link types. Links with higher immediacy more closely bind their arguments in time than links with lower immediacy.

I give an example of how the algorithm works with reference to Figure 5-1. The algorithm first constructs the graph shown, where the horizontal position of each event indicates where it was found in the text. The algorithm picks the first event annotated in the text, A, and walks forward to D along a *Begins* link. This produces the sequence A-D. It then walks backward from D along an *Immediately Before* link to B. Because *Immediately Before*, at 2, has a lower immediacy than *Begins*, at 3, B is inserted before A, giving the sequence B-A-D. The algorithm then walks forward from D to E along a simultaneous link, giving the sequence B-A-D-E. Finally, the algorithm walks backwards from E to C along a *Before* link. The algorithm walks back along the sequence until it finds a place where C is before all other more tightly bound events, producing the final sequence, C-B-A-D-E.

5.1.2 Semantic Roles

Once I constructed the event timeline, I assigned, if possible, a subject and an object to each event. I extracted this information from both the semantic role, referring expression, co-reference, and context relationship representations.

Every verb in the corpus was marked with a semantic role which gives the arguments to that verb represented as spans of text. Nearly every event in the corpus is associated, with at least one

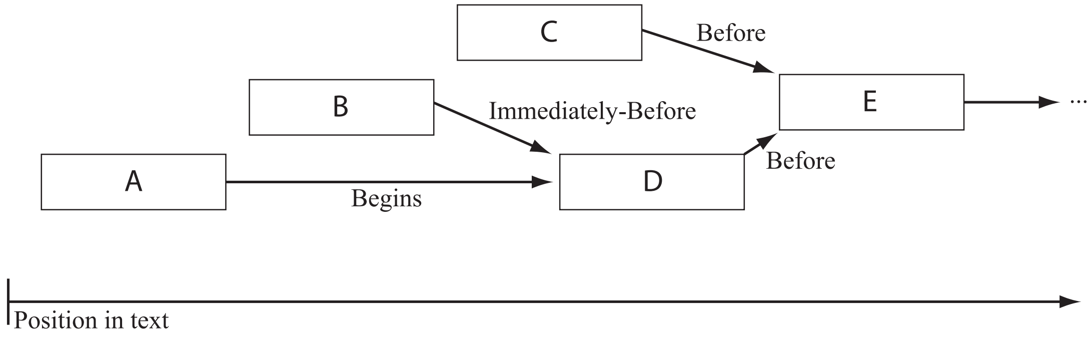


Figure 5-1: Example of link competition in event ordering algorithm. The ordering algorithm walks the graph in the order A, D, B, E, C. The correct final order of the events is C-B-A-D-E. Horizontal position of each box indicates where in the text the associated event was mentioned and each directed edge represents a TimeML time link annotation.

Argument	Description
ARG0	causal agent
ARG1	thing following
ARG2	thing followed

Table 5.2: Semantic roles for PropBank frame `follow.01`. This frame does not have ARG0-ARG1 subject-object structure, as the person performing the action is marked as ARG1 and the recipient is ARG2. This frame, rather, has ARG1-ARG2 subject-object structure.

semantic role via its verb expressions. In fact, of the 3,438 events on the tale timelines, only 2 events had no semantic role: these were verbs that had no arguments themselves, but were rather incorporated into other verbs as PropBank *Predicative* arguments. I manually specified the subjects and objects of these two events in post-processing. More often an event had more than one semantic role, meaning that the event was mentioned several times using a verb. I merged the subject and object fillers for each constituent semantic role, favoring the first-mentioned semantic role in case of a conflict.

I used each semantic role's associated PropBank frame to find the subject and object. According to PropBank conventions, the argument to the verb marked *ARG0* is usually the subject, and the argument marked *ARG1* is usually the object (see Table 4.2.2). Nevertheless, many PropBank frames do not have this ARG0-ARG1 subject-object structure because of the idiosyncracies of the frame definitions. For example, the roles for the frame `follow.01` are shown in Table 5.2. In the sentence "He followed her," *He* is marked ARG1 and *her* is marked ARG2, so this frame has an ARG1-ARG2 subject-object structure. Because this information is not encapsulated anywhere in PropBank, I manually classified the subject and object roles of all the PropBank frames found in the corpus.

Once the correct subject and object span was determined, I chose the largest referring expression inside each span as the most appropriate role filler.

Finally, I identified a class of verbs that were *symmetric* verbs, where the subject and object position were irrelevant for the semantics of the event. This included transitive verbs that implied an interaction among a group, such as in (1).

(1) They fought.

I manually classified PropBank frames as symmetric and this was taken into consideration when matching events in the merge step.

Tale No.	Tale Name	No. Words	No. Events	No. Times	Subor- dinate	Full Timeline	Filtered Timeline
148	Nikita the Tanner	646	104	8	22	75	16
113	The Magic Swan Geese	696	132	1	21	94	43
145	The Seven Simeons	725	121	4	28	87	42
163	Bukhtan Bukhtanovich	888	150	1	28	107	62
162	The Crystal Mountain	989	150	7	33	104	43
151	Shabarsha the Laborer	1202	236	8	85	122	55
152	Ivanko the Bear's Son	1210	223	20	143	143	65
149	The Serpent and the Gypsy	1228	250	12	111	138	80
135	Ivan Popyalov	1242	220	8	46	170	46
131	Frolka Stay-at-Home	1388	248	7	69	169	56
108	The Witch	1448	276	8	88	157	61
154	The Runaway Soldier...	1698	317	25	120	190	76
114	Prince Danila Govorila	1774	341	8	109	223	92
127	The Merchant's Daughter...	1794	331	7	89	234	89
140	Dawn, Evening and Midnight	1934	339	18	90	250	78
	Average	1257.5	229.2	9.5	67.3	150.9	60.3
	Sum	18862	3438	142	1009	2263	904

Table 5.3: List of tales in the corpus, their length in words, the number of events and time expressions on their full timelines, the number of subordinated events, and the length of the full and filtered timelines.

5.1.3 Referent Structure

Once I determined the referring expressions filling the subject and object roles of an event, it remained to substitute that referring expression with a one or more elementary referents. Sometimes this required replacing a composite referent with atomic referents, as in (2).

(2) Jack and Jill went up the hill. They fetched a pail of water.

In this example, the subject of the verb *fetched* is *They*, the set containing both Jack and Jill. I determined whether referents were atomic or composite by examining the context relation annotations. If a referent was composite, it was connected to its constituents via Set/Member context relationships.

In addition to groups of atomic referents, oftentimes the object of an action was a possession or body part of a character. I used context relations that indicated Owner/Possession, Body-Part/Body, and Part/Whole relationships to identify the appropriate referent. Given these context relationships, any composite referring expression could be replaced by a set of atomic referents.

5.1.4 Filtering the Timeline

The full timelines comprised 3,438 events and time expressions. As indicated in Table 5.4, there were only 276 function instances marked in the corpus. Therefore a majority of the objects on the timeline were irrelevant to structures I was trying to learn. I was able to circumscribe the set of events relating to the functions *a priori* by observing that the vast majority of Propp's functions are expressed as simple events denoted an interaction between two main characters. I therefore simplified the data by filtering out time expressions, subordinated events, any events not marked as an *Occurrence*, and any events that did not have both roles filled with *dramatis personae*. As can be seen in Table 5.3, removing these entries results in a nearly four-fold reduction in the number of objects that need to be considered, from 3,438 to 904.

5.2 Merging

Generating a linear timeline from each tale allowed me to construct an initial model, as described in Section 3.3.1. For this implementation of ASM, which I call ProppASM, I used a two-stage merge process with a highly-constrained prior distribution and greedy search. I used only the standard merge operation described by Stolcke and Omohundro (1994), the same as in Section 3.4.

Each stage had its own prior that incorporated a similarity function specific to that stage. Nevertheless, both priors were modulations on a basic geometric function on the number of states in the model. A geometric prior has a single parameter, the p value, which can take a value strictly between zero and one. For each stage the geometric function was modulated with a function penalizing models containing merged states violating that stage's similarity function. This is identical to the approach described in Sections 3.4.1 and 3.4.2 and is captured in Equations 5.1 and 5.2.

$$P(M) = p(1-p)^{n-1} \prod_i K(S_i) \tag{5.1}$$

$$K(S_i) = \begin{cases} 1 & \text{if } Sim(S_i) == true \\ t & \text{otherwise} \end{cases} \tag{5.2}$$

The symbol S_i stands for a state in the model, which may have been derived from merging two or more states in a previous model and thus may contain one or more events. Equation 5.1 defines the basic form of the prior for each stage, which is a geometric function with parameter p in the number of total states n, modulated by a penalty function K. Equation 5.2 defines K, which penalizes a model by a factor t for each state S_i that violates the similarity function for that stage. The penalty parameter may take any value from zero to one (inclusive at both ends). Thus the prior has two numerical parameters, p and t.

To design similarity functions that reproduce Propp's categories, it is important to consider the same features to which Propp himself was sensitive in his analysis. Propp actually describes in his work three features via which he found similarity between events: event semantics, *dramatis personae* involved, and the position of the event in the arc of the move. Consequently, I designed the two stages of ProppASM to exercise those aspects of similarity. The first stage merged together, roughly, non-generic events that were semantically similar and which involved compatible sets of *dramatis personae*. The second stage merged only states that contained more than one event, and of these states it merged those that were nearby with the same emotional impact and involving compatible sets of *dramatis personae*.

5.2.1 Stage One: Semantics

The similarity function for stage one was as follows. A state satisfied the similarity function if (1) all events in the state were non-generic, (2) each unique PropBank frame was represented at least twice, (3) all pairs of events in the state were synonymous or hyper-synonymous with regard to their Wordnet annotations, and (4) all pairs of events in the state had consistent sets of *dramatis personae*. By definition unmerged states satisfied the similarity function. I define these conditions in more detail below.

Generic Events Events whose verbs were marked with Wordnet senses falling into the lexicographer files of verbs of *communication*, *perception*, or *motion* were considered *generic*. These include verbs such as *say*, *see*, or *go*. They were excluded from merging because it was impossible to distinguish an informative, functional use of these words from a generic, filler sense. The verb *say* and its synonyms are a good example: these made up nearly three-quarters of all events and every one of Propp's functions included at least one *say* event. That is, characters could accomplish all of Propp's functions through speech acts. Characters could threaten each other (A, Villainy, or Pr, Pursuit), meet for the first time or offer assistance (D, First Encounter with the Donor), react to other's actions (E, Hero's Reaction to the Donor), offer one's services (C, Decision to Counteract), send a Hero on a quest (B Dispatch), and so forth.

Synonymity Two events were considered synonymous if their attached Wordnet senses, of the sense's hypernyms, shared synonyms. This defined a loose semantic similarity that allowed events to be clustered together on the basis of meaning.

Doubled PropBank Frames PropBank frames were attached to events through the semantic role annotation, as described in the previous section. The PropBank frame can be thought of as defining a specific type (as opposed to supertype) of an event. For a state to satisfy the similarity function, then, each PropBank frame found on an event in that state needed to be found on at least one other event in that state. This more specific semantic similarity served as a balance to the more generous similarity provided by Wordnet synonymity.

Consistent *Dramatis Personae* The *dramatis personae* involved in two events were considered consistent when they were either identical or proper subsets of each other. For both the subject and object arguments of an event, the *dramatis personae* tags for each participant in that argument was added to a set. If there was *Hero* tag was in the set, the *Helper* tag was also added, and vice versa. Two events were considered to have consistent *dramatis personae* if one event's set of elementary referents in the subject and object positions equaled (or was a proper subset of, or vice versa) the other event's set of elementary referents in the subject and object positions, respectively. If one of the events was marked as a *symmetric* event, where subject and object positions are irrelevant, the subject and object argument sets for each event were combined into one set for the purposes of matching.

As a practical matter, the implementation of this stage of ProppASM was split into two parts. The first part merged together non-generic events with matching PropBank frames that also had consistent *dramatis personae*. The second part took merged states produced in the first part and merged them on the basis of their synonyms or synonyms of their hypernyms. Given the size of the search space, greedy search was necessary to effect convergence for each part of this stage. Before the beginning of the merge process, pairwise state merges were tested to see if the states they produced violated the similarity function of that part. Then these pairwise merges were sorted so that merges consistent with the similarity function were considered first, followed by penalized merges. Merges were applied if they increased the posterior.

5.2.2 Stage Two: Valence

In second stage, a state satisfied the similarity function if (1) the valence across the events of the states matched, and (2) all pairs of events in the state had consistent *dramatis personae* distributions. As in stage one, unmerged states were considered by definition to satisfy the similarity function. Unlike in stage one, only states that already contained more than one event were considered for merging.

Matching Valence meant that the valences across the events in a state had to match. The possible valences for an event are shown in Table 4.6. The exception was that a *Neutral* valence was allowed to match any other valence.

This stage also took advantage of greedy search exactly as did the first stage, with the additional sorting of pairs according to how far apart the state's constituent events were, relatively, on their timelines. The position of each state was calculated as follows. The position of an event was defined as a fraction between 0 and 1, inclusive, corresponding to its relative position in its original linear timeline. The position of a merged node is the average position of its constituent events. Then pairwise merges were ranked according to the difference in position between the states they were merging, where the smallest differences were pushed to the front of the search queue.

5.2.3 Filtering

Finally, after the second stage, the model was filtered of states that did not contain at least four events. States with only three events were retained only if the events came from at least two different stories. The model produced by this filtering process was the model evaluated against the gold standard.

5.3 Gold Standard

I constructed the gold standard against which the final model was measured from the Propp function annotations produced as described in Section 4.2.6. The final set of function markings against which the ProppASM output was compared was actually much reduced from the list of functions in Propp's monograph for three reasons: Propp's omissions, functions not present or too sparse in the corpus data, and implicit functions.

Of the 31 functions identified by Propp, he did not identify where the first 7 functions occurred in the tales (these were the preparatory functions, marked with Greek letters). These functions therefore had to be excluded from the analysis. Of the remaining 24 functions, four functions, J, L, M, and N were not found in the 15 tales in my corpus, leaving 20 functions. Of these, an additional four, o, Q, Ex, and U had fewer than three instances. These were excluded from the target because I considered them too sparse to learn.

There were 276 function markings, of which 186 were explicit and 90 implicit. Because I did no commonsense inference, these implicit functions, or over 30% of the data, had no actual event instantiation in the text. This problem was largely circumvented by noting that the majority of the implicit functions were one of functions involved in the two pairs of E-F (Reaction and Receipt) and H-I (Struggle-Victory). In these cases, when one of a pair was implicit, the other would be explicit. For example, in the case of a Hero fighting with a Villain, only the actual fight was mentioned and the victory was left implicit, or the victory was mentioned and the fight left implicit. Thus for the purposes of measurement, I merged these two sets of functions together. This resulted in 231 explicit function markings out of 276; the remaining 45 implicit markings were excluded from the target. These data are summarized in Table 5.4.

As described in Section 5.1.4, the timeline was filtered before merging of events that were likely not relevant to the functions. While good, this filtering step did eliminate some actual function markings. As shown in Table 5.5, after filtering three functions, G, $\downarrow$ and T had only two instances each. Because ProppASM filtered out any merged nodes with fewer than three events in the final step, I did not expect that these functions would be learned by the algorithm. Nevertheless, these functions were included in the gold standard.

5.4 Analysis

I used four different procedures to analyze the performance of the ProppASM implementation and characterize its behavior. First is a measure of the overall quality of the clustering of events into Propp's functions. Second is individual F_1-measures for each of Propp's functions. Third is a measure of how well the transition structure was reproduced (the answer, not very well: see Section 5.5). Fourth is an examination of how well the implementation works with smaller amounts of data. I also examined the performance of the ProppASM implementation for different settings of the two parameters of the prior.

ProppASM achieved four clear successes. First, on a reasonable measure of event clustering, the algorithm performs fairly well, achieving on overall 0.714 chance-adjusted Rand index against Propp's original functions. Here I say "fairly well" because it is actually unclear how good this performance is: there is no prior work against which to compare. This is contrasted to when an experimenter might demonstrate an improvement of X percentage points over an earlier technique. This work, however, is the first attempt ever to learn Propp's functions from folktales by computer and so there is no previous technique against which to compare.

Second, the algorithm extracted the most central functions of the morphology, namely, the initial villainy A, the donor-encounter triplet DEF, the struggle with and victory over the villain HI, the liquidation of the villainy K, the Pursuit-Rescue doublet $Pr - Rs$, and the final reward W. These are all key functions, not only in the tales analyzed, but across Propp's morphology.

Third, the algorithm extracted three important functions, A (*Villainy*), W (*Reward*), and HI (*Struggle-Victory*) at individual F_1-measures of over 0.8.

Fourth, the algorithm exhibited a smooth degradation with smaller amounts of data, and is able to achieve respectable average chance-adjusted Rand index of 0.457 when examining just two stories.

Symbol	Description	Instances	Explicit
β	Absentation	0*	-
γ	Interdiction	0*	-
δ	Violation	0*	-
ϵ	Reconnaissance	0*	-
ζ	Delivery	0*	-
η	Trickery	0*	-
θ	Complicity	0*	-
A/a	Villainy/Lack	20	18
B	Mediation	7	7
C	Beginning Counteraction	13	7
$\uparrow$	Departure	14	13
D	Donor Encounter	19	16
E	Hero's Reaction	17	30
F	Receipt of Magical Agent	17	
G	Transference	5	4
H	Struggle	45	71
I	Victory	36	
J	Branding	0	-
K	Tension Liquidated	17	12
$\downarrow$	Return	14	10
Pr	Pursuit	27	18
Rs	Rescue	13	13
o	Unrecognized Arrival	$2^\dagger$	$2^\dagger$
L	Unfounded Claims	0	-
M	Difficult Task	0	-
N	Solution	0	-
Q	Recognition	$2^\dagger$	$2^\dagger$
Ex	Exposure	$1^\dagger$	$1^\dagger$
T	Transfiguration	3	$2^\dagger$
U	Punishment	$2^\dagger$	$2^\dagger$
W	Reward	12	0
Total‡		276	231

Table 5.4: Function counts in the data. *Unmarked by Propp; †Too few instances to train on; ‡Not including entries marked as having too few training instances

Symbol	Description	Explicit	Filtered
A/a	Villainy/Lack	18	15
B	Mediation	7	7
C	Beginning Counteraction	7	5
$\uparrow$	Departure	13	7
D	Donor Encounter	16	16
EF	Reaction & Receipt	30	29
G	Transference	4	2†
HI	Struggle & Victory	71	66
K	Tension Liquidated	12	9
$\downarrow$	Return	10	2†
Pr	Pursuit	18	14
Rs	Rescue	13	10
T	Transfiguration	3	2†
W	Reward	12	8

Table 5.5: Functions present in the corpus before and after timeline filtering. †Too few instances in data to expect extraction.

Method	Score
Strict	0.511
Interactive-only	0.581
Interactive-Non-Generics-Only	0.714

Table 5.6: Three different chance-adjusted Rand index measures of cluster quality. The scores range from most strict through most lenient.

5.4.1 Event Clustering

I used the chance-adjusted Rand Index (Hubert and Arabie 1985) to examine the quality of clustering of events into Propp's function categories. I created three measures which may be ranked, colloquially, from "strict" to "lenient." They were: (1) a *strict* score, where the clusters in the final model were compared against all Propp's clusters of explicit function markings as listed in *explicit* column in Table 5.5; (2) an *interactive-only* score, where the clusters in the final model were compared against Propp's explicit clusters with non-interactive events removed; and (3) an *interactive-non-generics-only* score, where the clusters in the final model were compared against Propp's explicit clusters with both non-interactive and generic events removed. These three results are listed in Table 5.4.1.

5.4.2 Function Categories

The second metric was the F_1-measure for individual function categories. Of the 14 function categories in the final data, 8 were recovered. These results are shown for the *interactive-non-generics-only* measure in Table 5.4.2.

The most striking success was the extraction of the *Struggle-Victory* combination function, HI. A full 51 instances were classified correctly, and, when measured against the filtered timeline, this resulted in an overall F-measure of 0.823. This success can probably be attributed to substantial uniformity of semantics for this particular function, in that all the verbs were verbs of competition and fighting.

The next notable success is the identification of A, villainy, and W, reward, functions with an F_1-measure of 0.8. These are two key functions because they bookend the action. Similar to HI, the semantic uniformity of these functions was important to their successful extraction. In Russian

Symbol	Description	Semantics	Hyp. Only	Both	Target Only	F_1-measure
A/a	Villainy/Lack	devour, drag, hurt, seize	3	12	3	0.8
D	Donor Encounter	drag, hit	0	6	10	0.545
EF	Reaction & Receipt	cover, eat, make	3	9	20	0.439
HI	Struggle & Victory	attack, break, cut, defeat, drag, fight, hit, hurt, push, race, seal, seize, throw, whistle	7	51	15	0.823
K	Tension Liquidated	fill	0	3	4	0.6
Pr	Pursuit	chase, chew	0	5	9	0.526
Rs	Rescue	strike, throw	1	6	4	0.706
W	Reward	gift, marry	1	6	2	0.8

Table 5.7: F_1-measures of identification of functions.

Method	Links only in Hypothesis	Links in Both	Links only in Target	F_1-measure
Strict	13	16	43	0.364
Interactive-only	10	19	35	0.458
Interactive-Non-Generics-Only	9	20	27	0.526

Table 5.8: Three F_1-measures for function-function transitions.

tales, the most common villainy is a kidnaping of a princess or another weak party. For the reward, it most commonly either marriage to the rescued princess or a gift of money.

5.4.3 Transition Structure

In addition to the grouping of events into functions I was also measured the quality of the extracted transition structure. The target morphology, corresponding to transitions between functions found in the 15 tales of the corpus, is shown in Figure 5-2(a). The structure extracted by the ProppASM implementation is seen in (b). There are several techniques to compare to finite state machines to obtain a quantitative measure of similarity. The most common in the grammar learning literature is to generate a large sample of traces from the target grammar, and see what fraction of those are accepted by the hypothesized grammar. This method, however, is not suitable for this stage of the research, because the alphabets of the target and learned grammars are so substantially different. As can be seen, the hypothesized grammar is missing 6 of 14 symbols. This means that any trace from the target that contains any of those symbols will perforce not be accepted by the hypothesis.

A more informative metric is an F_1-measure on individual links in the graphs. This method results in the data shown in Table 5.4.3.

5.4.4 Data Variation

I ran the algorithm over subsets of the stories in the corpus to investigate performance degradation with smaller amounts of data. Figure 5-3 shows the performance of ProppASM, measured by the three chance-adjusted Rand index measures as in Table 5.4.1, at the optimal parameter values over different subsets of the corpus. Each data point in the graph is an average of all n-sized subsets of stories from the folktale corpus. As can be seen, ProppASM's performance drops off smoothly, until, when only two stories are being considered at a time, it retains a surprisingly good value of 0.457 for the No-Generics measure, 0.360 for the Interactive-Only measure, and 0.325 for the Strict measure. This measurement shows that the implementation is actually quite robust to variation in the data.

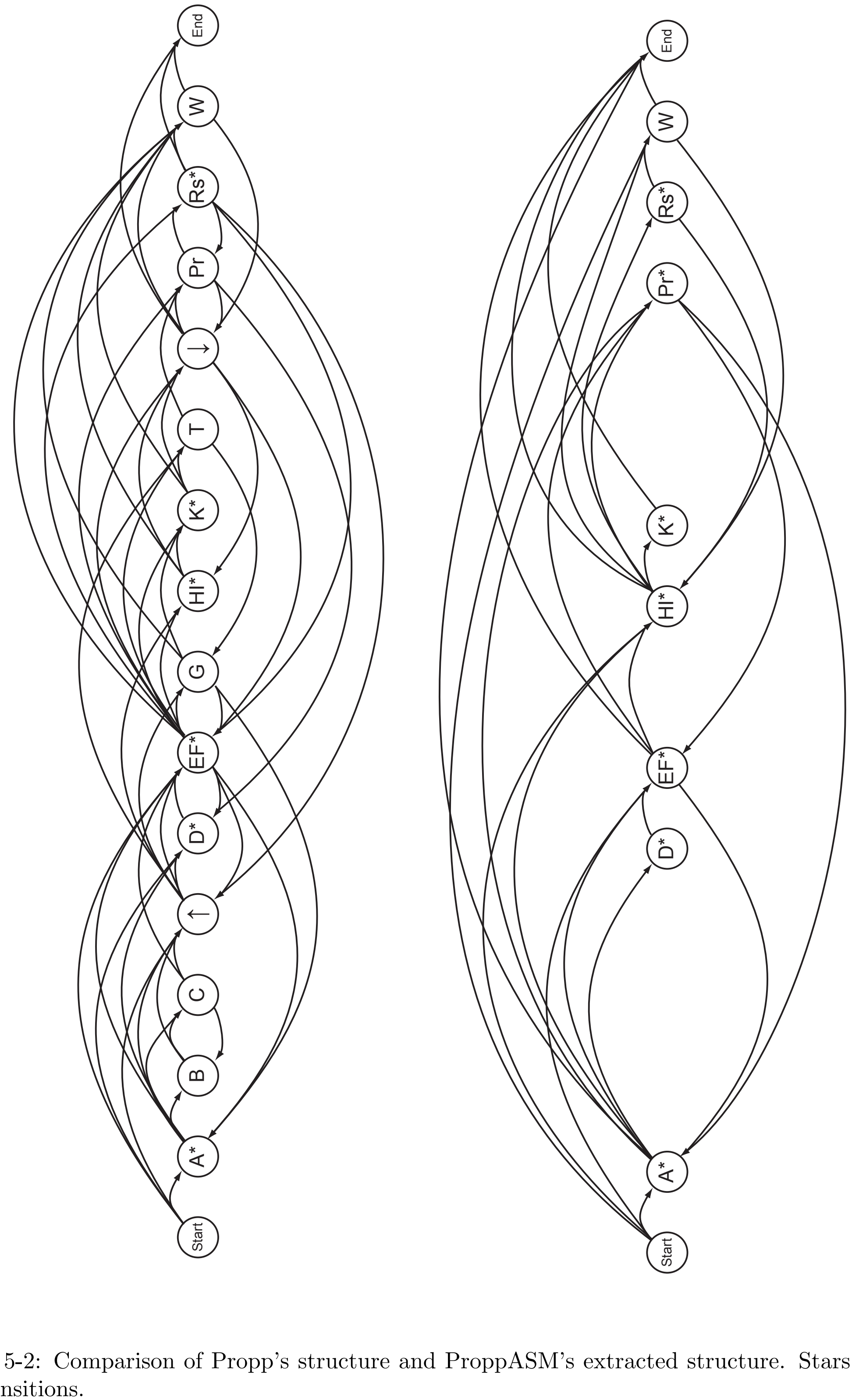


Figure 5-2: Comparison of Propp's structure and ProppASM's extracted structure. Stars indicate self-transitions.

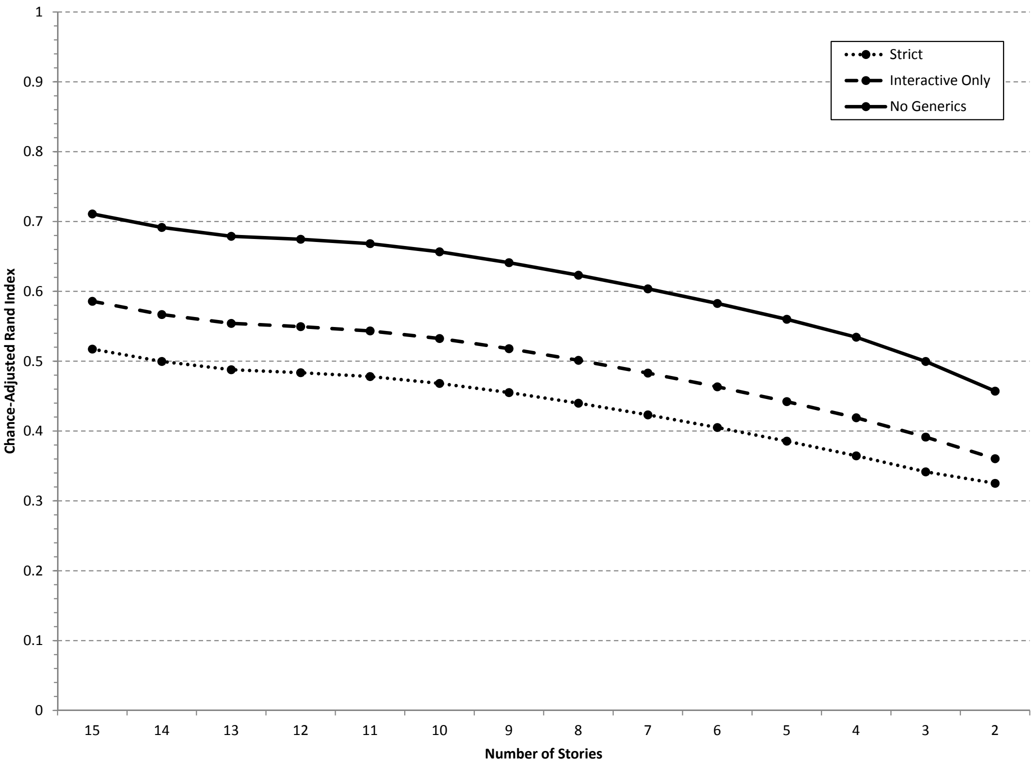


Figure 5-3: Performance of the ProppASM implementation on all subsets of the corpus.

5.4.5 Parameter Variation

In addition to running ProppASM at its optimal parameter settings, I also investigated the implementations's performance at different penalties and geometric p-values. Figures 5-4 and 5-5 show the performance of the algorithm, as measured by the three chance-adjusted Rand index measures, for different values of the geometric prior p-value.

Figure 5-4 shows a p-value sweep from $ln(p) = -1 \times 10^{-2}$ to $ln(p) = -1 \times 10^{-50}$, with a penalty of zero ($t = 0$). The penalty depends on the p-value of the prior, in that the penalty must not be so severe that it penalizes good merges. Roughly, the upper-bound of the penalty is inversely proportional to the p-value. The largest penalty allowed was calibrated for a particular p-value by raising the penalty from zero until the graph showed a narrow peak. This final calibration is shown in Figure 5-5, where for the optimal p-value, namely $ln(p) = -1 \times 10^{-21}$, the largest penalty allowed was $ln(t) = -50$.

5.5 Shortcomings

While the results produced by the ProppASM implementation did meet or exceed my expectations in several aspects, it fell short in other areas. A careful analysis of the shortcomings of the implementation will allow an accurate judgement of the quality and generality of the result, and to chart the way forward to achieving even better performance.

5.5.1 Performance

While overall the clustering of events into functions was fairly good, especially as measured by the most lenient measure, and the F_1-measures for five of the eight functions extracted were above 0.6,

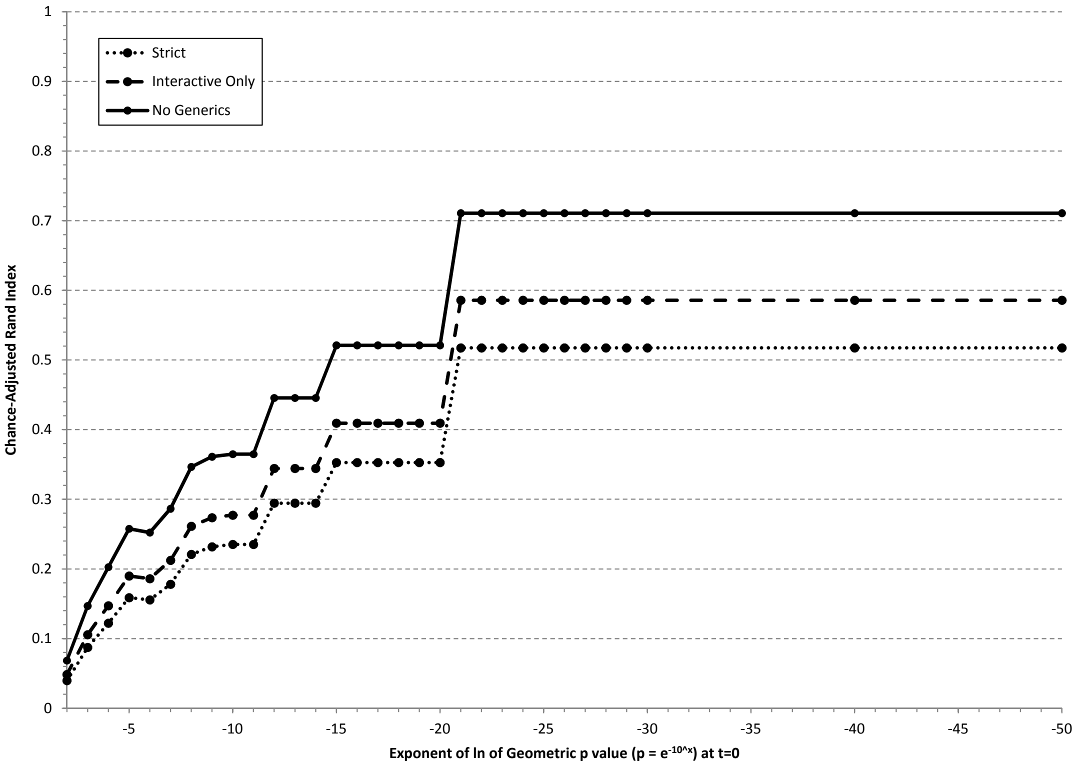


Figure 5-4: Performance of the ASM algorithm for different p-values at zero penalty.

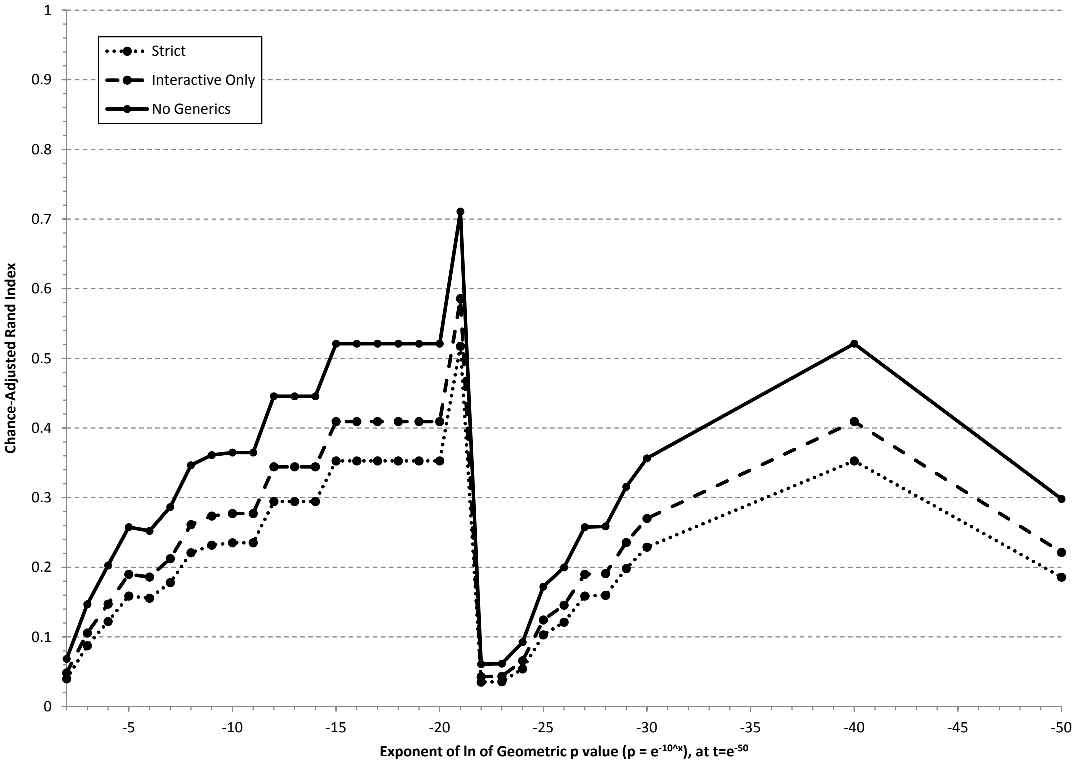


Figure 5-5: Performance of the ASM algorithm for different p-values at $t = e^{-50}$.

there were still six function categories from the data which were not identified at all. These were B, C, $\uparrow$, G, $\downarrow$, and T.

As shown in Table 5.5, functions G, $\downarrow$, and T had only two instances each in the final filtered data used by the algorithm; it is therefore not surprising that the algorithm did not find these, as we specifically limited the implementation to finding groups of three or more. This leaves three functions, B, C, and $\uparrow$, that seemed to have sufficient presence in the data, but were still not identified. B is called the Mediation, and can be thought of as when the Dispatcher (the king, the father) asks the Hero to rescue the Princess. Of the seven instances in the final data, four are expressed as *say*, and one each as *commission*, *plead*, and *reach* (as in "the King's pronouncement reached the Heroes."). Thus five of these fall into our generic verbs of communication (*say* and *plead*), which were explicitly ignored in this implementation. C, the decision by the Hero to take up the call to action, and $\uparrow$, the departure from home, both suffer from similar problems, in that they are both preferentially expressed by generics of either communication or movement.

Another failure of the algorithm is in extracting the transition structure of the grammar. The F_1-measure over transitions is rather poor on the strict scale, 0.364, and rises only to 0.526 on the lenient scale when non-interactive and generic events are excluded from the target. This inability to accurately extract the transitions leads to a few strange omissions in the transition graph, most notably the lack of a direct transition between Pr and Rs, *Pursuit* and *Rescue from Pursuit*.

In fact, the lesser performance on the transitions, relative to the clustering, is to be expected. The transition structure is fully determined by a specified clustering. While the quality of the clustering depends only on whether individual events are put in the correct groups, the quality of the extracted transitions depends on whether individual events that are adjacent in a story are simultaneously classified correctly. Therefore if the algorithm has, say, a 50% chance of putting any particular event in the correct function category, then the probability of putting a pair of events both into the correct category is 25%. Thus the quality of the transitions will always trail that of the clustering. Nevertheless, this indicates that there is hope for overall improvement of the transition extraction if one can only improve the overall quality of the clustering of events into function categories.

5.5.2 Prior

Another major area of concern is the design of the prior. The structure of the prior is what determines whether or not ASM converges. There are at least three criticisms of the prior underlying the ProppASM implementation: scale-dependence, reliance on *dramatis personae* annotations, and lack of generality.

Scale-dependence is probably the least worrisome of these criticisms. By scale-dependence I mean that the optimal p-value and penalty for the ProppASM prior as described will be dependent on the size of the initial model. As the size of the initial model increases, the optimal p-value will go up, approaching 1. For a prior which uses a zero penalty ($t = 0$), a p-value that works for a certain model size will work for smaller model sizes, as is shown in Figure 5-3, where the size of the initial model decreases with the size of the data subset.

What is needed is a prior whose parameters scale not with the size of the initial model, but rather scale with the number of nodes expected in the final, filtered model. This is because no matter how many stories we add to our initial model, the number of final clusters should remain approximately the same. Doubling the number of stories does not double the number of functions expected. In fact, above some number of stories, one expects the number of functions to be constant.

Correcting this problem is mostly a matter of engineering and experimentation. I chose to base the prior on a geometric distribution because of its simplicity, but a more sophisticated distribution, such as the Beta function or a Chinese Restaurant Process (Pitman 2006) might be more suitable.

A more dire criticism of the ProppASM prior is its reliance on the *dramatis personae* annotations. This annotation captures a significant intellectual contribution of Propp's morphology, and is absolutely critical to extracting the function categories. This annotation is also implicit in the Event Valence representation, where each event is marked for its emotional impact on the Hero (rather than any other character). For the ProppASM procedure to be made completely culture- and theory- independent, it must be engineered so as to be able to extract the *dramatis personae* categories at the same time as the function categories. This transforms the problem from a single clustering problem over events to a co-clustering problem over both events and referents.

Early on in development of the described implementation of ProppASM I did attempt this co-clustering. Experiments indicated that it was easy to pare down the total number of referents in the stories to a relatively tight superset of referents that took *dramatis personae* roles. This filtering could be done, first, by removing non-agentive, non-atomic referents from the set, and then removing referents not mentioned at least a certain number of times (between 3 and 5 times seemed to be the right cutoff). I then attempted to use non-parametric clustering techniques (Kemp et al. 2006) to cluster these referents into categories resembling *Hero*, *Helper*, *Villain*, and so forth. After quite a bit of experimentation, however, I was unable to find a combination of features that reproduced the *dramatis personae* categories, and I eventually returned to the more pressing task of demonstrating the feasibility of the function clusters. Clearly, discovering the *dramatis personae* categories directly from the data will need to be addressed if the technique is to be considered to extract independently Propp's morphology from the stories.

The final criticism of the prior is that it has embedded in it strong assumptions about what features govern the similarity of events: event semantics, *dramatis personae* participants, and position in the story. In the end it was my choice of those three features that allowed ProppASM to extract the function categories. My choice was motivated by Propp's discussion of his technique, but this still begs the question: Where do these features come from? One would like a prior that was a bit more theoretically agnostic, or features that were more clearly motivated by a theory of narrative more general than, but including, Propp's, or perhaps features that fall out of a theory of cognitive pattern discovery.

Furthermore, because the prior has such strong assumptions embedded, it is unclear how well it would port to other domains involving narrative structure. Given the generality of the features chosen, relative to folktales, I expect that the prior will work well on other collections of folklore and other areas where Lehnert-style plot units are the important structures. But how well would the prior port to other domains which look nothing like folktales? There is an especially clear problem with the current implementation if the domain contains nothing analogous to the *dramatis personae* categories. And it is clear to me that this particular prior is wholly inappropriate for extracting something like Levi-Strauss' theory of narrative structure (Lévi-Strauss 1955), where the relevant similarities are diachronic rather than synchronic. If the work is to truly be applied across multiple domains and find relevant and interesting structure in a scientifically robust way, the structure of the prior will need to be examined and justified in much greater detail than I have so far done.

5.5.3 Representations

There are several deficiencies in the representational suite used as a basis for the ProppASM implementation. Previously mentioned is the undesirability of relying on the *dramatis personae* annotation and the related Event Valence representation. However, there are numerous areas where the information made available to the ProppASM implementation does not approximate the "surface semantics" that an individual will extract from a story and have available to use for doing a Proppian-style analysis in a satisfactory way.

The most obvious failing is the lack of information that allows the algorithm to handle generic verbs. A good example of this is the generic verb *say*. Nearly three-quarters of all the events in these tales were verbs of saying, and many of them were marked as functions. There needs to be a method of identifying the purpose and function of these speech acts aside from their direct semantic character. Numerous other generic verbs suffer from the same problem. Information that ties these generic events into the plan, goal, or causal structure of the story is needed to improve the extraction of functions expressed in that way.

Another example of information that would be useful would be important or salient locations. I did not annotate any information relating to the location or movement of characters other than as expressed in the verbs. Nevertheless, there are usually just a small number of important places between which the characters move. In these tales, the locations of Home versus Away, or real, upper world versus magic, underworld are important distinctions. When the Hero makes a transition between one of these two worlds it is almost invariably correlated with an important function. Having this information available could potentially improve extraction of functions such as β, $\uparrow$, $\downarrow$, o, and Pr.

Finally, information that would allow implicit function occurrences to be discovered would be a useful improvement to the technique. For this we need an ability to fill in implicit events in the story timeline in a robust way with good agreement. This is akin to marking reader inferences that are commonly made. To my knowledge, however, no computational linguistic technique or annotation scheme has been developed to do this. This remains, then, an area for future research.

5.5.4 Gold Standard

A final class of criticisms questions whether Propp's analysis is really the ground truth; if Propp's analysis does not reflect the actual content of the tales, then it would be difficult to reproduce completely. Is his morphology the ground truth? While hard to quantify without more data, my assessment, after my detailed study of the tales in the corpus, is that while he looks to have gotten the morphology right in broad outlines, there are definitely functions he missed. This includes categories he identified but did not mark in particular tales, and categories that seem to be present in the tales but were not picked out by Propp. The most striking example of the latter was that there seems to be a repeated element, that perhaps one may call "Deceit by the Clever Hero," that is found in at least five tales of my corpus. I discovered this missing function quite by accident when I printed out the timelines of the tales annotated with Propp's function markings. I noted that several tales had long stretches of action in which Propp did not identify any functions. Across these barren stretches are shared what looks like, for all purposes that I can imagine, a missing Proppian function.

Chapter 6

Related Work

In this chapter I review alternative approaches to the ones pursued in this work. First, I review several different theories of narrative structure and show how Propp's analysis was best suited to the goals of this work. Second, I review some recent approaches to automatically extracting narrative structure. Finally, I review alternative tools and representations that might have been used for the corpus annotation.

6.1 Structural Analyses of Narrative

Theories of narrative structure abound. They come from literary analysis, folklore, anthropology, cognitive science, and artificial intelligence. I identify five different classes of theories of narrative structure and I examine them in turn to show why morphologies in general, and Propp's theory in particular, was the best choice for this work. I evaluate each class along three dimensions. First, the theory should be well enough specified to provide a specific target for annotation and learning. Second, the theory should be relatively well vetted and generally accepted by the relevant experts. Third, the theory should be culturally sensitive because one main goal of this work is to understand culture as exposed through stories.

6.1.1 Universal Plot Catalogs

Perhaps the theory that comes immediately to mind when discussing narrative structure is Joseph Campbell's well known theory of the monomyth (Campbell 1949). The theory is influential across popular culture, has been the subject of widely popular documentaries and books (Campbell and Moyers 1991) and has been consciously applied by authors and screenwriters (Vogler 1992), even being consciously incorporated into structure of the *Star Wars* movie trilogy.

Campbell identified three sections to a myth: Separation, Initiation, and Return. Across these three sections are spread 17 stages, listed in Table 6.1. According to Campbell, myths may contain all, some, or just one of these stages, and the order is generally free.

Departure					
	Separation		Initiation		Return
1.	The Call to Adventure	6.	Road of Trials	12.	Refusal of the Return
2.	Refusal of the Call	7.	Meeting with the Goddess	13.	Magic Flight
3.	Supernatural Aid	8.	Woman as Temptress	14.	Rescue from Without
4.	First Threshold	9.	Atonement with the Father	15.	Return Threshold
5.	Belly of the Whale	10.	Apotheosis	16.	Master of Two Worlds
		11.	The Ultimate Boon	17.	Freedom to Live

Table 6.1: The three sections of the monomyth, and its 17 stages.

Despite its widespread popularity, the monomyth theory is the easiest narrative theory to dismiss. It is based on highly doubtful psychological proposals as well as scant anthropological and folkloristic evidence.

First, Campbell claims an underlying pre-cultural psychology which begets the monomyth form. It relies heavily on Freudian psychoanalysis and Jungian theories of archetypes, both traditions quite suspect from a modern cognitive point of view.

Second, Campbell makes wildly broad claims of universality. He claims, for instance, that all stories are instances of the monomyth:

> Whether we listen with aloof amusement to the dreamlike mumbo jumbo of some red-eyed witch doctor of the Congo, or read with cultivated rapture thin translations from the sonnets of the mystic Lao-tse; now and again crack the hard nutshell of an argument of Aquinas, or catch suddenly the shining meaning of a bizarre Eskimo fairy tale: *it will be always the one, shape-shifting yet marvelously constant story that we find*. . . . (Campbell 1949, p.3)

Such an extraordinary claim requires equally extraordinary evidence, but Campbell provides nothing more than anecdotes. Folklorists attempting to address the gap find there is no evidence of a monomyth form in all cultures (Dundes 2005). It would even be fair to say that it has been conclusively demonstrated that, far from the monomyth being the only form, it is only one among many common story forms (Kluckhohn 1959). While modern folklorists admire Campbell for his ability to fire the popular imagination, they consider his theories sorely lacking when it comes to describing the facts on the ground (Segal 1978).

Campbell's theory is unsatisfactory for a third reason: it proposes no way to handle cultural differences, one of the main motivations of this work. Even if the monomyth theory were true, it is clear there are differences across cultures with regard to folktale form and content. If we are to explain these, we must start with a theory which admits cultural difference, not papers over it.

While Campbell's work is the most prominent example, works of that flavor abound in the popular literature. Authors will claim seven (Booker 2004), twenty (Tobias 1993), thirty-six (Polti 1924) or more "basic plots" that, they claim, describe one, most, or all stories. None that I have examined have a solid grounding in a scientific understanding of cognitive, psychological, and cultural principles. Inevitably, all extant examples suffer from the same flaws: lack of evidence and cultural invariance.

6.1.2 Story Grammars

Story grammars was a push to formalize narrative structure and document its psychological effects that was directly inspired by Propp's monograph. Chomsky's advances in the theory of grammars and languages provided new tools that could be applied to artifacts other than sentences, and those interested in narrative structure eagerly took up the charge. A good example of such an effort is Prince's monograph *A Grammar of Stories* (Prince 1973). He proposes terminal and non-terminal symbols representing generic events or states, and proposed a set of rules that he used to "parse" several forms of story into their constituent elements. Work in cognitive science showed related interests: David Rumelhart (1975) wrote down a grammar of "problem-solving stories," after which Jean Mandler and Nancy Johnson (1977) attempted to expand that grammar to handle simple folktales, and used it to inform different cognitive psychology experiments.

Story grammars focused on relatively universal aspects of stories: stories are made up of events; stories tend to have a global structure that might loosely be called preparation, action-reaction, and conclusion; and stories can be nested (recursion). Story grammars generally ground out in unnamed "events." While there was some controversy regarding whether or not grammars were the right representation for the task (Black and Bower 1980), story grammars were actually used to good effect in describing and explaining various effects during story recall

Story grammars look quite similar to Propp's theory and therefore are an attractive possible target for learning. However, while they have some interesting psychological evidence in their favor, their major shortfall is they do not have any account of cross-cultural differences. They claim, much like universal plot catalogs (albeit with more precision and more convincingly) a general psychological story structure shared across all people. The second problem is that story grammars are generally not specific enough to make for an actual target of a learning, in that there is too large a gap between the most specific element in the grammar (the generic "happening" or "event") and the specific events we must manipulate in annotated data.

6.1.3 Scripts, Plans, & Goals

Roger Schank and Robert Abelson (1977) proposed the theory of *scripts* that, while not intended directly as a narrative theory, serves just as well. Rather than focusing on rewrite rules and transformations for representing episodic structure and embedding, the script represented plan-goal structure directly. As a theory of narrative structure, it was somewhat more specific than the plot catalog and story grammar approaches: it proposed specific knowledge structures and representations that accounted for commonsense rules, defaults, and inferences in common situations. Unfortunately, other than providing one or two compelling examples, Shank and Abelson left the compilation of script catalogs as future work.

Lehnert took the idea of the script and specialized it into the idea of plot units, which, as I discussed in Section 3.4.3, is almost directly analogous to Propp's knowledge structures.

Despite the promise of scripts and plot units, there was never a proposal that focused on a specific culture or the scripts involved in a specific, natural set of stories. A specific catalog of plot units, targeted at a specific culture, would have been a suitable replacement for Propp's morphology as a learning target. Because they did not provide such a catalog the script approach falls short as target for learning.

6.1.4 Motifs & Tale Types

I would be amiss if I did not mention the two original classification schemes for folktales. These schemes do not necessarily qualify as *theories* of narrative structure, but they have had a huge impact on understanding and studying folktales, and they are also accompanied by vast catalogs of structures indexed to different cultures and regions (Uther 2004), a rare feature indeed. These two schemes are the motif and the tale-type.

Stith Thompson defined the *motif* as "the smallest element in a tale having a power to persist in tradition." (Thompson 1946) Whether the elements identified by the standard motif indexes actually fit this description is open to question, but there is no doubt that the identified motifs are found across a number of tales. A motif is a repeated element in a set of tales. A motif generally falls into one of three categories: it may be a character, such as a god, unusual animals, monsters, or a conventional person (such as the youngest child or the cruel stepmother); it may be an item of context, such as a magical item, an unusual custom, or strange belief; and finally, it may be a notable incident. A motif index lists all tales that contain a motif, but a tale being grouped under a particular motif does not imply any *genetic* relationship between the listed tales: those tales so grouped may or may not be variants of the same tale or have overlapping ancestry.

On the other hand, a *tale type* is a collection of traditional tales that group together on the basis of their motifs in a sort of family resemblance. In contrast to motifs, grouping into a tale type does imply a genetic relationship (i.e., later tales are evolutions or variations of earlier tales). Tale types are often given the name of a well-known tale that falls into that type, e.g., *Cinderella* or *Snow White*.

While motif and tale type indexes are of great use to the professional folklorist, and thus they meet our criterion of being reflective of folklore reality, it has been much discussed why both of these categorization schemes are useful only for indexing purposes (Dundes 1962). Indeed, Propp cited the shortcomings of motifs and tale types for categorizing and dissecting folktales as one motivation for developing his morphology (Propp 1968, Chapter 1).

As an aside, learning motif indices could be an interesting first step toward formalizing the categorization of tales by motif. There is clearly some relationship between motifs and functions, in that certain motifs are likely correlated with certain functions for a particular culture. An investigation of this relationship would be an interesting piece of future work.

6.1.5 Other Morphologies

We are left, then, with morphologies. As this work demonstrates, Propp's morphology is clearly specific enough to be a target of annotation and learning. It also respects the observable facts with regards to folkloristics. What about its cultural sensitivity?

Certainly Propp's morphology is itself specifically a morphology of the *Russian* fairy tale: Propp himself states this. Despite this, Propp notes that even though the morphology is specifically aimed at Russian fairy tales, it does seem to be an effective summary of not only the tales he analyzed but many outside his corpus (Kluckhohn 1959; Fischer 1963). Importantly, however, he does not make a claim of universality.

From Propp's original method and results, it can be seen how morphologies, as a general method, may be applied to different cultures to yield different functions, subtypes, transition structure, and *dramatis personae*. There are at least two examples of repetitions of the generation of a morphology from original stories and they give us interesting insights into what cross-cultural variations we might expect when Propp's method is applied to other cultures.

The first morphology was by Alan Dundes (Dundes 1964), for North American Indian folktales. While it is not a complete morphology, it is outlined in enough detail that we can see both similarities and differences as compared with Propp. He confirms that North American Indian tales may have multiple moves and identifies ten widespread functions (which he calls *motifeme* patterns) that are quite similar to some of Propp's functions. I discuss eight of these overlaps in the following list.

- **Lack** (L): Directly analogous to Propp's function a, *Lack*, except that in American Indian tales not only a lack but an *over-abundance* may cause problems. Interestingly, Dundes does not mention an explicit villainy pattern independent of an induced lack or abundances in the tales he analyzes. Pairs with LL.

- **Lack Liquidated** (LL): Directly analogous to Propp's K, *Lack Liquidated*. One difference here is that often LL occurs directly after L without any intervening functions - this is not a structure found in Propp's morphology. Pairs with L.

- **Task** (T): Might be considered analogous to Propp's function M, *Difficult Task*. Pairs with TA.

- **Task Accomplished** (TA): Might be considered analogous to Propp's N, *Solution*. Pairs with T.

- **Interdiction** (Int): Directly analogous to Propp's function γ, but also usually occurs after the motivating tension, L. Pairs with $Viol$.

- **Violation** ($Viol$): Directly analogous to Propp's δ, but more often found intervening between L and LL, rather than in the preparatory sequence. Pairs with Int.

- **Deceit** (*Dct*): Most directly similar to Propp's function in the preparation, but this is probably more analogous to an unidentified Deceit/Deception by the Hero/Helper. Pairs with *Dcpn*.

- **Deception** (*Dcpn*): Again, similar to Propp's η, but more likely analogous to a missing function in Propp's morphology that might be called "Deceit by the Protagonist." Pairs with *Dct*.

The second example of a carefully constructed morphology for a culture other than Russian is Benjamin Colby's analysis of Eskimo tales (Colby 1973). Colby's analysis bears great similarity to Propp's except that his sample was far smaller, only fourteen tales. Colby makes at least two innovations of interest to the construction of morphologies. First, he groups functions (which he calls *eidons*) into primary and secondary groups. The primary group appears to be relatively strictly ordered; the secondary eidons serve as modifiers to the primary eidons and are less strictly ordered. The secondary eidons have similarities with several of Propp's functions, for example, *Departure of the Hero* (Propp: ↑, Colby: Dp).

Second, following Lakoff's analysis (Lakoff 1972), Colby introduces grammar rules that lie between the content-sensitive gross structure move level and the intermediate regular grammar of the functions. Colby calls these *categorical components* and they have names such as *value motivation*, *immediate motivation*, *preliminary action*, *main action* and so forth. These intermediate grammar rules, Colby notes, are useful in accounting for several of the reversals of function order that are observed in his folktale sample. He notes that, while Propp did not identify them, rules of this sort almost certainly operate on Propp's folktales as well. This is an encouraging innovation because it deals cleanly with one of the most unsatisfying aspects of Propp's theory, namely, the unexplained local inversions of order in some of his tales. However, it is also worrisome because introducing these intermediate production rules elevates the grammatical complexity of the intermediate level from regular to context-free, with a consequent increase in difficulty of learning.

With regard to the identities of the functions themselves, Colby notes the following: "Five of Propp's functions—villainy, departure, struggle, victory, and return—are similar in content to Eskimo eidons and have the same title. They differ from the Eskimo units, however, in their sequential meaning in terms of the overall narrative action and in their content at a lowers level." (Colby 1973, p.645) This means that, much like Dundes's Native American morphology, Colby's Eskimo morphology shares some number of functions with Propp's, but the overlap is not exact and the order is somewhat different. Interestingly, the subtypes of the functions are substantially different.

From these two examples it would seem that a rigorous repetition of Propp's method on a different culture's folktales produces an analysis that is both similar to and different from Propp's morphology of Russian tales. Roughly, the similarities are found in the gross structure and also the somewhat overlapping identities of functions. But the function identities are not exactly the same and their order is varied. At the fine structure level, there are are substantial differences in function subtypes even for the case of exactly analogous functions.

Such analyses reassures that Proppian-style morphologies were the right theory for this work. Indeed, contrasting and comparing these three morphologies gives additional motivation for pursuing a computerized version of Propp's, Dundes's, and Colby's analysis technique. The observation that the morphologies are most similar in the upper structure, somewhat similar at the intermediate level, but substantially different in their fine structure is interesting and potentially important. How can we be sure that these different morphologies accurately reflect the state of the world? Dundes's and Colby's morphologies were done in full light of Propp's analysis. How can we be sure they were not polluted by it, and the similarities are just spurious? To answer this question, we need to be able to reproduce their analysis automatically.

6.2 Extracting Narrative Structure

There has been some recent interesting work on learning narrative structures. I review two important techniques.

First, Nathanial Chambers and Dan Jurafsky (2008, 2009) leverage distributional learning over very large corpora to identify common sequences of events. The technique relies on a pointwise mutual information score between verbs that share arguments to build up common pairs of events and their orders. These pairs are then knitted together to form narrative chains. An example narrative chain that might be extracted is shown in Table 6.2.

row	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_3$	$\mathbf{s}_4$
1	∅	walk into restaurant	∅	enter restaurant
2	∅	∅	walk to the counter	go to counter
3	∅	find the end of the line	∅	∅
4	∅	stand in line	∅	∅
5	look at menu	look at menu board	∅	∅
6	decide what you want	decide on food and drink	∅	make selection
7	order at counter	tell cashier your order	place an order	place order
8	∅	listen to cashier repeat order	∅	∅
9	pay at counter	∅	pay the bill	pay for food
10	∅	listen for total price	∅	∅
11	∅	swipe credit card in scanner	∅	∅
12	∅	put up credit card	∅	∅
13	∅	take receipt	∅	∅
14	∅	look at order number	∅	∅
15	∅	take your cup	∅	∅
16	∅	stand off to the side	∅	∅
17	∅	wait for number to be called	wait for the ordered food	∅
18	receive food at counter	get your drink	get the food	pick up order
19	∅	∅	∅	pick up condiments
20	take food to table	∅	move to a table	go to table
21	eat food	∅	eat food	consume food
22	∅	∅	∅	clear tray
22	∅	∅	exit the place	∅

Figure 6-1: Example script for eating at a restaurant, after (Regneri, Koller, and Pinkal 2010, Figure 2).

Events		Roles
	A search B	A = Police
	A arrest B	B = Suspect
	A plead C	C = Plea
D acquit B	D convict B	D = Jury
	D sentence B	

Table 6.2: Example Chambers-Jurafsky narrative chain, after (Chambers and Jurafsky 2008, p.602).

Narrative chains have several interesting points of commonality and difference with this work. We are both trying to identify chains of events commonly found across sets of texts. Furthermore, their work is another data point supporting the argument that knowing the roles of the characters (e.g., who's the Protagonist) is critically important for identifying common narrative structure. Divergently, the technique relies on an incredible weight of texts (they train on the GigaWord corpus, using over 1M texts) to find similarities; this is in contrast to ASM, which works passably well on a mere two tales. Second, the narrative chain model sits quite close to the meaning of the texts: verbs are considered identical when they share root forms. With ASM I go beyond this surface representation to abstract and generalize from the data — for example, using semantic knowledge to unify items like kidnap and seize and then further unify these with a verb like *torment* to achieve a "harm" or "villainy." Importantly, too, the narrative chain method does not learn cycles, an important part of the narrative structure of folktales.

The second related work, by Michaela Regneri and colleages (Regneri, Koller, and Pinkal 2010) seeks to learn event scripts from lists of actions. The technique is a variation of the multiple sequence alignment technique from bioinformatics. They were able to extract reasonable script-like structure from the data. Differences from my work include the type of data (subject-generated lists of key actions in performing a task versus natural stories) and, like the Chambers and Jurafsky work, Regneri's work cannot learn cycles. There is also no attention paid to filtering out unimportant events, as their starting the data contained only events relevant to the script. An example of a script extracted by their technique is shown in Figure 6-1.

6.3 Corpus Annotation

I briefly examine alternatives to the tools, formats, and representations used for the corpus annotation.

6.3.1 Tools & Formats

Using a programmed tool to assist in annotation is not a new idea. It is worthwhile to briefly mention several intellectual precursors and potential alternatives to the Story Workbench.

First, there is the class of data formats, usually coupled with a data management API. The primary examples of these include TEI, the Text Encoding Initiative; MASC, the Manually-Annotated Subcorpus; and IBM's UIMA, Unstructure Information Management Architecture. I will discuss each in turn.

TEI in a long-standing data format (Burnard and Bauman 2008) cast in SGML or XML, which some might see as a competitor to the data format of the Story Workbench. A standard example of a TEI-encoded text is shown in Figure 6-2.

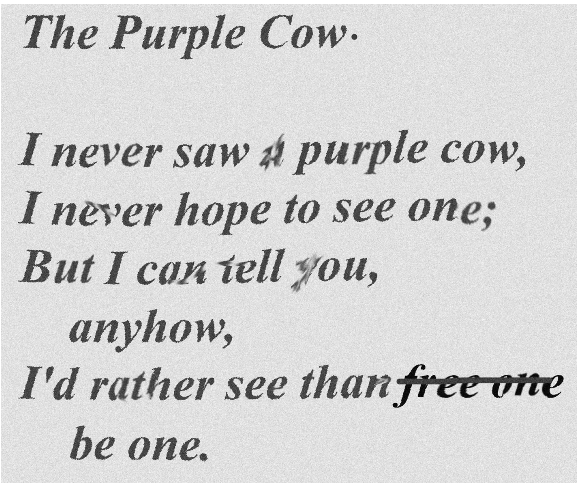
The primary reason not to use TEI is that it is not a stand-off format. Rather, it is *inline*, with the annotations interspersed with the text and other annotations. A stand-off format is critical to having a modular and extensible annotation architecture. Without stand-off annotations, one is limited to inserting annotations in the text. If we are to use XML for this, as TEI recommends, one is limited to hierarchal structures, a severe constraint. There are many examples of representations in the Story Workbench that do not properly nest with each other. For example, each TimeML temporal link annotation covers two events. While sometimes the two events are in the same sentence, more often than not they are in different sentences. Therefore, there is a nesting conflict: one cannot annotate both sentences and temporal links in the same text if using the standard TEI format. If one instead implement stand-off annotation in TEI, one is immediately required to write an extension to TEI. One would be merely choosing TEI for its use of a few top level tags and XML, which is not a good trade-off. Therefore, I did not use TEI as the file format.

MASC, the Manually-Annotated Sub-corpus, is a recent effort to bring multi-representational corpus annotation to the mainstream (Ide et al. 2010). This is an interesting effort and has many commonalities with the data that I have produced. The proximal reason for not using the MASC GrAF annotation format as a backing format for the Story Workbench data structure was its late entry into the field. By the time MASC was available I had already created the Story Workbench. On the downside, the MASC annotation format is quite loose in its data types and does not include much of the infrastructure provided the Story Workbench, such as dependencies between representations, factories, and so forth. Its format is also verbose, making it difficult to understand the structure of the file at a glance.

IBM's Unstructured Information Management Architecture, or UIMA, is perhaps the most serious competitor to the underlying Story Workbench data model and format (Apache 2011). UIMA was developed contemporaneously with the Story Workbench, and so it was not available when I was making the decision whether or not to create my own annotation tool. However, at UIMA's current advanced stage of development, it would possibly serve as a good substitute for the data storage and processing backend. It has many of the same data model features, such as stand-off annotation, use of generally accepted standards such as XML, being based on Java and the Eclipse application framework, a strong annotation type system, and annotation factories. Attractively, it also already has an established user base and is seeing some adoption. Indeed, it has a few advantages over the Story Workbench data structures: it has plugable "flow controllers" which allow a developer to customize the order in which annotation factories are run; it allows multiple versions of a text to be included in a document; and it permits artifacts others than text to be annotated. It may be the case that I will replace the Story Workbench data model with UIMA in a future release.

Although the three formats and tool chains discussed above, MASC, UIMA, and TEI, might be considered competitors for the backend of the Story Workbench, they do not replace the front end, which is critical to the annotation efforts.

Aside from these formats, there is a class of tools one might call "Annotation MATLABs,"



The Purple Cow·

I never saw a purple cow,
I never hope to see one;
But I can tell you,
  anyhow,
I'd rather see than ~~free one~~
  be one.

```
<TEI.2>
<teiHeader>
  <fileDesc>
    <titleStmt>
      <title>The Purple Cow</title>
      <author>Frank Gelett Burgess</author>
    </titleStmt>
    <publicationStmt>
      <p>Text entry by <name>Walter Underwood</name> on
      <date>1999-02-01</date>. This transcription is
      in the public domain.</p>
    </publicationStmt>
    <sourceDesc>
      <p>From the <title>The Lark</title>, San Francisco, <date>1896</date>.</p>
    </sourceDesc>
  </fileDesc>
</teiHeader>
<text>
  <body>
    <div1 type="poem">
      <head>The Purple Cow</head>
      <lg>
        <l>I never saw <supplied>a</supplied> purple cow,</l>
        <l>I never hope to see one;</l>
        <l>But I can tell <supplied>y</supplied>ou, anyhow,</l>
        <l>I’d rather see than <del>free one</del> be one.</l>
      </lg>
    </div1>
  </body>
</text>
</TEI.2>
```

Figure 6-2: Example of a text fragment and its TEI-encoded equivalent.

the primary examples of which are GATE (Cunningham 2002) and Ellogon (Petasis et al. 2002). The most important difference between these platforms and the Story Workbench is that they are designed for use by researchers and are not suitable for use by non-technical annotators. They have complicated user interfaces that take a significant amount of training and background knowledge to use. These contrast with the Story Workbench, which smooths the annotation process by hiding all the complex machinery behind an intuitive and user-friendly graphical user interface.

6.3.2 Representations

Because the number of representations that could have been used in the annotation effort is quite large, I will not go through all the possible alternatives here in detail. In general, representations were chosen to cover the semantics of what I expected to be important for learning the morphology. The particular choices were influenced primarily by three considerations: first, solid linguistic foundations; second, coverage and vetting with a large corpus; and third, implementation difficulty. I will evince three examples of alternative representations that were not chosen because of one or more of these considerations.

The first example is the representation of temporal order of the events in the text. The two main contenders for this role were the TimeML link representation (Pustejovsky et al. 2003) and Allen's temporal logic (Allen 1983). I chose TimeML first because of coverage: Aspectual and temporal links in TimeML can be fully represented Allen's temporal relations, but the subordinative relations are not covered by Allen's representation. Second, I chose TimeML because it was backed up by a large corpus of annotated examples, wherein it was found to cover many of the important phenomena. Finally, using TimeML allowed me to take immediate advantage of another representation from the TimeML suite, events.

The second example of a representational choice is of marking arguments to verbs. Here the main competitors were PropBank (Palmer, Kingsbury, and Gildea 2005) and FrameNet (Fillmore, Johnson, and Petruck 2003). Both are supported by substantive annotated corpora and both are well motivated linguistically. Here, practical considerations won out, in that there has been much work on statistical semantic role labelers and I used these to great effect in seeding the annotations for the annotators, saving quite a bit of time.

The final example is Malec's *pftml* markup language for Propp's functions (Lendvai et al. 2010). Primarily theoretical considerations led to the development of my own markup for Propp's functions, in that the pftml scheme does not provide a way to annotate the presence of implicit functions. This was crucial. Another important difference was my scheme's ability to handle not only the region of impact of the function but also identify the defining verb for each function. Beyond these theoretical considerations, there was the practical challenge of integrating pftml into the Story Workbench data format, which would have sent the whole tool back to the drawing board.

Chapter 7

Contributions

This work has four main contributions.

7.1 Analogical Story Merging

The primary contribution of this work was the development and demonstration of *Analogical Story Merging* (ASM), an algorithm for extracting narrative structure from sets of annotated stories. I used ASM to learn a substantive portion of Vladimir Propp's influential theory of the structure of folktale plots from an annotated subset of the actual tales he used in his analysis.

The challenge was to take descriptions at one semantic level, namely, happenings in the world as they are communicated in folktales, and abstract them to the next higher level: structures such as *Villainy*, *Conflict*, and *Reward*. To do this, ASM builds upon the machine learning technique of Bayesian Model Merging for learning regular grammars. ASM moves beyond Bayesian model merging by introducing two modifications. First, it allows states in the model to have internal structure that can inform decisions as to whether or not to merge two states via an analogical mapping algorithm. Second, it adds a filtering step at the end of the merge process where unmerged states are discarded and highly merged states become symbols in the final inferred grammar. I demonstrate that, despite ASM's large search space, the size of which is defined by Bell's number, a carefully-tuned prior allows the algorithm to converge.

To learn Propp's morphology, I used a two-stage merge process. The first stage merged semantically similar, non-generic events involving the same character classes, and the second stage merged compound states created in the first stage based on their position in the story and emotional valence. These two stages captured three important features that influenced Propp's identification of his event clusters: the semantic nature of the events, the types of characters involved, and the position of the events in the story.

The behavior of both stages was induced by a prior specific to each stage in which a geometric distribution over the size of the model, heavily biased toward a small number of states, was modulated by a function that penalized non-preferred merges. A greedy search algorithm was sufficient for the algorithm to converge in a reasonable amount of time.

7.2 The Story Workbench

I developed the *Story Workbench*, a generic text annotation tool to create the data on which the ASM algorithm was to be run. The Story Workbench is cross-platform, user-friendly, and highly extensible. It fills a major gap in corpus linguistics in which software support for corpora is usually *ad hoc*, poorly designed and documented, and not flexible enough to be adapted to future studies.

The Workbench complies with three basic design principles, namely: (1) Build on top of popular tools with large user and developer communities; (2) Use open-source and freely-available programming libraries; and (3) Adopt widely-used and well-documented standards. Furthermore, as much as possible I strove to satisfy five desiderata in its design: (1) The annotation storage format should be human-readable and tool-independent; (2) The annotation storage format should be modular and extensible; (3) The tool framework should be highly functional, modular, and extensible; (4) The tool should not commit to a specific operating system; and (5) The tool should not commit to specific NLP technologies.

The Story Workbench also supports, with specific features, at least nine activities spread across the three main stages of the annotation process of (i) Individual Annotation, (ii) Double Annotation, and (iii) Annotation Development: (1) It supports the automatic creation of annotations via its data model and annotation factories; (2) It supports annotation problem identification and mitigation with build rules and resolutions; (3) It facilitates inspection of annotations with detail views; (4) It enables manual creation and correction of annotations via creator views; (5) It facilitates comparing and contrasting annotations from different annotators via meta representations and the merge tool; (6) It supports annotation adjudication via build rules, detail views, and creator views; (7) It enables tool updates and annotation text distributions via a source control and update framework; (8) It eases annotation scheme implementation with a representation versioning system; and (9) It supports identification of annotation scheme problems and annotation process evaluation with the inter-annotator agreement tool. Each one of the features is fully extensible, allowing developers to add their own functionality and tools to the Workbench through a plugin architecture.

I implemented 18 representations that could be semi-automatically annotated by human annotators that range across the spectrum of syntax and semantics. They are (1) Tokens, (2) Multi-word Expressions, (3) Sentences, (4) Part of Speech Tags, (5) Lemmas, (6) Word Senses, (7) Context-Free Grammar Parses, (8) Referring Expressions, (9) Referent Attributes, (10) Co-reference Bundles, (11) Time Expressions, (12) Events, (13) Temporal Relationships, (14) Context Relationships, (15) Semantic Roles, (16) Event Valence, (17) Propp's *Dramatis Personae*, and (18) Propp's Functions.

7.3 Annotated Russian Folktale Corpus

I created a corpus of annotated Russian folktales, which is the largest and most comprehensively annotated narrative corpus assembled to date. The corpus comprises 15 tales drawn from the 100 tales that Propp originally analyzed, in particular, 15 out of 21 of his single-move tales — tales that contained just a single complication-liquidation arc. The corpus contains 18,862 words, and was fully annotated by 12 annotators across 18 representations using the Story Workbench over the course of a year. Each representation was doubly-annotated and adjudicated at inter-annotator F_1-measures ranging 0.54 to 0.98, but clustering around 0.7 to 0.8.

Of the 18 representations annotated, five of them were developed *de novo* specifically for this work: Referent Attributes, Referent Links, Event Valences, Propp's *Dramatis Personae*, and Propp's Functions. The function representation is especially notable because it goes above and beyond all previous attempts to translate Propp's function scheme into a modern representation that can be annotated with high inter-annotator agreements across a large corpus.

7.4 Reproduction of a Substantive Portion of Propp's Morphology

I demonstrated that Analogical Story Merging, when run with the annotated folktales in the Russian folktale corpus, was able to reproduce a substantive portion of Propp's morphology.

I assessed the algorithm's performance over the data in four different ways. First, I measured the quality of clustering events into Propp's original categories via the chance-adjusted Rand index against three interpretation of the original data: a strict measure that compared against all the original data, an in-between measure that measured against only interactive events, and a loose measure that considered only interactive events of a non-generic nature. These measures were 0.511, 0.581, and 0.714, respectively, where the Rand index runs from -1 to 1. The progression of these measures show that future work must concentrate on inferring or annotating additional information for generic events that will allow their similarity to other events to be assessed more accurately.

Second, I computed the F_1-measure of each inferred category against the matching Propp category. These numbers ranged from 0.439 to 0.823. However, three extremely important categories are identified with F-measures above 0.8, namely *Villainy*, *Struggle-Victory*, and *Reward*.

Third, I computed the F_1-measure of category-to-category transitions, also against three interpretations of the data, *strict*, *interactive-only*, and *interactive-non-generic-only*. This was the worst performing measure, with results of 0.364, 0.458, 0.526, respectively.

Fourth, I determined that the performance of the algorithm degraded smoothly when run over smaller subsets of the data. When run over only two stories, the algorithm was still able to achieve a reasonable averages measures of 0.325, 0.360, and 0.457 for the three different chance-adjusted Rand index clustering measures.

References

Afanas'ev, Aleksandr. 1957. *Narodnye russkie skazki.* Moscow: Gos. Izd-vo Khudozh. Lit-ry.

Agirre, Eneko, and Philip Edmonds. 2007. *Word Sense Disambiguation.* Dordrecht, The Netherlands: Springer-Verlag.

Allen, Johnathan. 1983. "Maintaining knowledge about temporal intervals." *Communications of the ACM* 26 (11): 832–843.

Apache, UIMA Development Community. 2011. "UIMA Tutorial and Developers' Guides, Version 2.3.2." Technical report, The Apache Software Foundation, Forest Hill, MD.

Bartlett, Frederic. 1920. "Some experiments on the reproduction of folk-stories." *Folklore* 31 (1): 30–47.

———. 1932. *Remembering: A Study in Experimental and Social Psychology.* Cambridge: Cambridge University Press.

Berend, Daniel, and Tamir Tassa. 2010. "Improved bounds on Bell numbers and on moments of sums of random variables." *Probability and Mathematical Statistics* 30 (2): 185–205.

Bird, Steven, and Mark Liberman. 2001. "A formal framework for linguistic annotation." *Speech Communication* 33 (1-2): 23–60.

Black, John, and Gordon Bower. 1980. "Story understanding as problem-solving." *Poetics* 9 (1-3): 223–250.

Booker, Christopher. 2004. *The Seven Basic Plots: Why We Tell Stories.* London: Continuum Press.

Bray, Tim, Jean Paoli, Michael Sperberg-McQueen, Eve Maler, and Francois Yergeau. 2006. "Extensible Markup Language (XML) 1.0, Fourth Edition." Technical report, World Wide Web Consortium, Cambridge, MA.

Burnard, Lou, and Syd Bauman. 2008. *TEI P5: Guidelines for Electronic Text Encoding and Interchange.* Oxford: Technical Council of the TEI Consortium.

Campbell, Joseph. 1949. *The Hero with a Thousand Faces.* Princeton, New Jersey: Princeton University Press.

Campbell, Joseph, and Bill Moyers. 1991. *The Power of Myth.* New York: Anchor Books.

Carletta, Jean. 1996. "Assessing agreement on classification tasks: the Kappa statistic." *Computational Linguistics* 22 (2): 249–254.

Chambers, Nathanael, and Daniel Jurafsky. 2008. "Unsupervised learning of narrative event chains." *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL).* Columbus, OH, 789–797.

———. 2009. "Unsupervised learning of narrative schemas and their participants." *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL).* Suntec, Singapore, 602–610.

Colby, Benjamin. 1973. "A partial grammar of Eskimo folktales." *American Anthropologist* 75 (3): 645–662.

Cunningham, Hamish. 2002. “GATE, a general architecture for text engineering.” *Computers and the Humanities* 36 (2): 223–254.

Díaz-Agudo, Belén, Pablo Gervás, and Federico Peinado. 2004. “A case based reasoning approach to story plot generation.” *Proceedings of the 7th European Conference on Case Based Reasoning (ECCBR).* Madrid, Spain, 142–156.

Dundes, Alan. 1962. “From etic to emic units in the structural study of folktales.” *Journal of American Folklore* 75 (296): 95–105.

———. 1964. *The Morphology of North American Indian Folktales.* Volume 195. Helsinki: Folklore Fellows Communications.

———. 2005. “Folkloristics in the twenty-first century.” *Journal of American Folklore* 118 (470): 385–408.

Falkenhainer, Brian, Kenneth Forbus, and Dedre Gentner. 1989. “The Structure-Mapping Engine: Algorithm and examples.” *Artificial Intelligence* 43 (1): 1–63.

Fellbaum, Christiane. 1998. *Wordnet: An electronic lexical database.* Cambridge, MA: MIT Press.

Fillmore, Charles, Christopher Johnson, and Miriam Petruck. 2003. “Background to Framenet.” *International Journal of Lexicography* 16 (3): 359–361.

Finlayson, Mark, and Patrick Winston. 2006. “Analogical retrieval via intermediate features: The Goldilocks hypothesis.” Technical report MIT-CSAIL-TR-2006-071, MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA.

Fischer, John. 1963. “The sociopsychological analysis of folktales.” *Current Anthropology* 4 (3): 235–295.

Gamma, Erich, and Kent Beck. 2004. *Contributing to Eclipse: Principles, Patterns, and Plug-ins.* Boston: Addison-Wesley.

Gildea, Daniel, and Daniel Jurafsky. 2002. “Automatic labeling of semantic roles.” *Computational Linguistics* 28 (3): 245–288.

Gold, Mark. 1967. “Language identification in the limit.” *Information and Control* 10 (5): 447–474.

Gosling, James, Bill Joy, Guy Steele, and Gilad Bracha. 2005. *The Java Language Specification (Third Edition).* Upper Saddle River, NJ: Addison-Wesley.

Halpin, Harry, Johanna Moore, and Judy Robertson. 2004. “Automatic Analysis of Plot for Story Rewriting.” *Proceedings of the Conference on Experimental Methods in Natural Language Processing (EMNLP).* Barcelona, Spain, 127–133.

Hervás, Raquel, and Mark Finlayson. 2010. “The prevalence of descriptive referring expressions in news and narrative.” *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL).* Uppsala, Sweden, 49–54.

Higuera, Colin de la. 2010. *Grammatical Inference: Learning Automata and Grammars.* Cambridge: University of Cambridge Press.

Hofstede, Geert. 1980. *Culture's Consequences: International Differences in Work-Related Values.* Beverly Hills: Sage Publications.

Hubert, Lawrence, and Phipps Arabie. 1985. “Comparing partitions.” *Journal of Classification* 2 (1): 193–218.

Ide, Nancy, Collin Baker, Christiane Fellbaum, and Rebecca Passonneau. 2010. “The Manually Annotated Sub-Corpus: A Community Resource For and By the People.” *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL).* Uppsala, Sweden, 68–73.

Kemp, Charles, Joshua Tenenbaum, Thomas Griffiths, Takeshi Yamada, and Naonori Ueda. 2006. “Learning systems of concepts with an Infinite Relational Model.” *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI).* Boston, MA, 381–388.

Klein, Daniel, and Christopher Manning. 2003. “Accurate unlexicalized parsing.” *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL).* Sapporo, Japan, 423–430.

Kluckhohn, Clyde. 1959. “Recurrent themes in myths and mythmaking.” *Daedalus* 88 (2): 268–279.

Kulkarni, Nidhi, and Mark Finlayson. 2011. “jMWE: A Java Toolkit for Detecting Multi-Word Expressions.” *Proceedings of the 8th Workshop on Multiword Expressions.* Portland, OR, 122–124.

Lakoff, George. 1972. “Structural complexity in fairy tales.” *The Study of Man* 1 (2): 128–150.

Lehnert, Wendy. 1981. “Plot units and narrative summarization.” *Cognitive Science* 5 (3): 293–331.

Lendvai, Piroska, Thierry Declerck, Sandor Daranyi, Pablo Gervás, Raquel Hervás, Scott Malec, and Federico Peinado. 2010. “Integration of Linguistic Markup into Semantic Models of Folk Narratives: The Fairy Tale Use Case.” *Proceedings of the 7th Language Resources and Evaluation Conference (LREC).* Malta, 1996–2001.

Lévi-Strauss, Claude. 1955. “The structural study of myth.” *The Journal of American Folklore* 68 (270): 428–444.

Mandler, Jean, and Nancy Johnson. 1977. “Remembrance of things parsed: Story structure and recall.” *Cognitive Psychology* 9 (1): 111–151.

Marcus, Mitchell, Mary Marcinkiewicz, and Beatrice Santorini. 1993. “Building a large annotated corpus of English: the Penn treebank.” *Computational Linguistics* 19 (2): 313–330.

Omohundro, Stephen. 1992. “Best-first model merging for dynamic learning and recognition.” *Advances in Neural Information Processing Systems* 4:958–965.

Palmer, Martha, Paul Kingsbury, and Daniel Gildea. 2005. “The Proposition Bank: An annotated corpus of semantic roles.” *Computational Linguistics* 31 (1): 71–105.

Petasis, Georgios, Vangelis Karkaletsis, Ggeorgios Paliouras, Ion Androutsopoulos, and Constantine Spyropoulos. 2002. “Ellogon: A New Text Engineering Platform.” *Proceedings of the Third International Conference on Language Resources and Evaluation.* Las Palmas, Canary Islands, Spain, 72–78.

Pitman, Jim. 2006. *Combinatorial Stochastic Processes.* Berlin: Springer-Verlag.

Polti, Georges. 1924. *The Thirty-Six Dramatic Situations.* Franklin, OH: James Knapp Reeve.

Pradhan, Sameer, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Daniel Jurafsky. 2005. “Support vector learning for semantic argument classification.” *Machine Learning* 60 (1-3): 11–39.

Prince, Gerald. 1973. *A Grammar for Stories.* The Hague: Mouton.

Propp, Vladimir. 1968. *Morphology of the Folktale.* Austin: University of Texas Press.

Pustejovsky, James, Jose Castano, Robert Ingria, Roser Sauri, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. “TimeML: Robust specification of event and temporal expressions in text.” *Proceedings of the 5th International Workshop on Computational Semantics (IWCS).* Tilburg, The Netherlands.

Regneri, Michaela, Alexander Koller, and Manfred Pinkal. 2010. “Learning script knowledge with web experiments.” *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL).* Uppsala, Sweden, 979–988.

Rota, Gian-Carlo. 1964. “The number of partitions of a set.” *The American Mathematical Monthly* 71 (5): 498–504.

Rumelhart, David. 1975. Pages 211–236 in *Notes on a schema for stories*, edited by Daniel Bobrow and Allan Collins, Language, Thought, and Culture: Advances in the Study of Cognition. Berkeley: Academic Press.

Saurí, Roser, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. "TimeML Annotation Guidelines, Version 1.2.1." Technical report, Brandeis University Linguistics Department.

Schank, Roger, and Robert Abelson. 1977. *Scripts, Plans, Goals and Understanding.* Northvale, NJ: Lawrence Erlbaum and Associates, Inc.

Schunn, Christian, and Kevin Dunbar. 1996. "Priming, analogy, and awareness in complex reasoning." *Memory and Cognition* 24 (3): 271–284.

Segal, Robert. 1978. "Joseph Campbell's theory of myth." *Journal of the American Academy of Religion* 44 (suppl. 1): 97–114.

Seifert, Colleen, Robert Abelson, Gail McKoon, and Roger Ratcliff. 1986. "Memory connections between thematically similar episodes." *Journal of Experimental Psychology: Learning, Memory, and Cognition* 12 (2): 220–231.

Spencer-Oatey, Helen. 2000. *Culturally Speaking: Managing Rapport through Talk across Cultures.* New York: Continuum Press.

Stolcke, Andreas, and Stephen Omohundro. 1994. "Inducing probabilistic grammars by Bayesian model merging." *Grammatical Inference and Applications*, Volume 862 of *Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 106–118.

Tenenbaum, Joshua, Charles Kemp, Thomas Griffiths, and Noah Goodman. 2011. "How to grow a mind: Statistics, structure, and abstraction." *Science* 331 (6022): 1279–1285.

Thompson, Stith. 1946. *The Folktale.* New York: Holt, Rinehart and Winston.

Tobias, Ronald. 1993. *20 Master Plots.* Writer's Digest Books: Blue Ash, OH.

Toutanova, Kristina, Daniel Klein, Christopher Manning, and Yoram Singer. 2003. "Feature-rich part-of-speech tagging with a cyclic dependency network." *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL).* Edmonton, Canada, 252–259.

Uther, Hans-Jörg. 2004. *The Types of International Folktales: A Classification and Bibliography. Based on the system of Antti Aarne and Stith Thompson.* Volume 284286. Helsinki: Folklore Fellows Communications.

Valiant, Leslie. 1984. "A theory of the learnable." *Communications of the ACM* 27 (11): 1134–1142.

van Rijsbergen, Cornelius. 1979. *Information Retrieval.* Oxford: Butterworth-Heinemann.

Vogler, Christopher. 1992. *The Writer's Journey: Mythic Structure for Storytellers and Screenwriters.* Studio City, CA: M. Wiese Productions.

Winston, Patrick. 1980a. "Learning and reasoning by analogy." *Communications of the ACM* 23 (12): 689–703.

———. 1980b. "Learning and reasoning by analogy: The details." Technical report AIM-520, MIT Artificial Intelligence Laboratory.

———. 2011. "The strong story hypothesis and the directed perception hypothesis." Edited by Patrick Langley, *Proceedings of the AAAI Fall Symposium on Advances in Cognitive Systems.* AAAI Press, 345–352.

Young-Lai, Matthew. 2009. "Grammar inference." Edited by Ling Liu and M. Tamer Özsu, *Encyclopedia of Database Systems.* Berlin: Springer-Verlag, 1256–1260.